VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Návrh a realizace intranetové aplikace pro správu majetku ve veřejné správě

Design and Implementation of Property Management Intranet Application in Public Administration

Student: Marek Pasz

Vedoucí bakalářské práce: Mgr. Martin Skýba

Ostrava 2013

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Marek Pasz**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R001 Aplikovaná informatika

Téma: Návrh a realizace intranetové aplikace pro správu majetku ve veřejné správě
Design and Implementation of Property Management Intranet Application in Public Administration

Zásady pro vypracování:

1. Úvod
2. Teoretická východiska tvorby intranetové aplikace
3. Analýza současného stavu správy majetku
4. Návrh a realizace intranetové aplikace
5. Zhodnocení navrhovaného řešení
6. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

BÖHMER, Marian. *Zend Framework: Programujeme webové aplikace v PHP*. Brno: Computer Press, 2010. ISBN 978-80-251-2965-4.
SCHNEIDER, Robert D. *MySQL: oficiální průvodce tvorbou, správou a laděním databází.* Praha: Grada, 2006. ISBN 80-247-1516-3.
PONKRÁC, Miloslav. *PHP a MySQL bez předchozích znalostí.* Brno: Computer Press, 2007. ISBN 978-80-251-1758-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Martin Skýba**

Datum zadání: 23.11.2012
Datum odevzdání: 10.05.2013

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

Prohlašuji, že jsem celou práci, včetně všech příloh, vypracoval samostatně.

V Ostravě dne 8. 5. 2013

Marek Pasz

Poděkování

V první řadě bych rád poděkoval svému vedoucímu bakalářské práce panu Mgr. Martinu Skýbovi za rady a čas, věnovaný při průběžných konzultacích. Děkuji také Ing. Michaele Karasové za poskytnuté materiály, které značně pomohly při zpracovávání práce.

Obsah

1. Úvod

Informační systémy a informační technologie s nimi spojené jsou v dnešních podnicích a organizacích velmi často využívané pro jejich možnosti ulehčení práce s informacemi. Informační systémy pomáhají jak při sběru informací, tak i při jejich zpracovávání apod. Pro jejich vícestranné využití se dnes používají jak v podnikové sféře, tak ve sféře veřejné správy.

V případě Městského úřadu v Českém Těšíně, konkrétně majetkoprávního oboru, se pro požadované informace (např. smlouvy) muselo chodit do archivu, kde bylo nutno danou smlouvu hledat a až pak s ní šlo dále pracovat. Při této činnosti byl problémem zdlouhavý přístup k informacím a taky zdlouhavé posílání smlouvy poštou, ať už na jakýkoli orgán veřejné správy, nebo jinde, pokud byla vyžádána. Pro vyřešení těchto problémů, ale i dalších, vznikla myšlenka vytvoření aplikace pro správu majetku (tj. smluv a s nimi spojených nemovitostí, osob atd.)

Práce je rozdělená do jednotlivých částí. V první části se nachází teorie pro obeznámení s technologií, se kterou se v aplikaci pracuje. Jelikož aplikace využívá webového rozhraní spolu s databází, pak jsou v této části více přiblížené právě tyto technologie. Další část se zabývá analýzou stavu správy majetku, konkrétně na Městském úřadě v Českém Těšíně. Je tam uvedený nejen současný stav zpracování informací a žádostí, ale taky požadavky na změnu nebo možnosti zlepšení. V další části je obsažen popis a metody postupu návrhu a realizace této konkrétní aplikace. Při tomto popisu se vychází z požadavků zaměstnanců. V poslední části je hodnoceno navrhované řešení, jeho klady, zápory, přínosy i nedostatky.

K vytvoření dané intranetové aplikace je využito zázemí Zend frameworku, který využívá kombinace programovacího jazyka PHP a MySQL. Tento framework umožňuje využívat, jako i ostatní frameworky, jeho naprogramované knihovny sloužící k rychlejšímu a snadnějšímu programování webových aplikací. Tato forma programování je v dnešní době značně oblíbená z důvodu velké flexibility a zároveň připravených k použití, často se opakujících, částí webových aplikací. K programování je používáno aplikace NetBeans, která podporuje integraci Zend knihovny, takže nejen, že dokáže udržovat strukturu kódu Zend frameworku, ale taky dokáže napovídat jednotlivé části knihovny, a proto je práce při vytváření aplikace značně rychlejší.

Cílem této bakalářské práce je tedy návrh a vytvoření intranetové aplikace pro správu majetku na Městském úřadě v Českém Těšíně, konkrétně na majetkoprávním odboru. Pomocí aplikace je poskytována pracovníkům na MěÚ správa a kontrola nad smlouvami, nemovitostmi, osobami, a možnosti jejich vzájemného užití. V neposlední řadě je pomocí aplikace upozorňováno pracovníky MěÚ na termíny a jejich úkoly, a taky vytvořená sjednocená platforma pro správu informací. Výstupy dané aplikace jsou poskytovány jak ve formě webových stránek, tak i ve formátu PDF dokumentů, což je jeden z požadavků zaměstnanců MěÚ. Dílčím cílem práce je aplikaci realizovat pouze lokálně (intranet) z důvodu, že je navržena výhradně pro pracovníky MěÚ a ne pro veřejnost. Dalším dílčím cílem je zajistit bezpečnostní stránku aplikace. Tedy obecným cílem práce je umožnit zaměstnancům MěÚ přechod ze starých metod udržování informací o městském majetku k moderní intranetové aplikaci pro správu majetku.

2. Teoretická východiska tvorby intranetové aplikace

2.1. Internet

Internet je veřejná síť, která umožňuje komunikaci, sdílení nebo předávání informací mezi zařízeními připojenými k této síti. Internet se v konečném důsledku skládá z jednotlivých lokálních sítí (LAN), u kterých vzájemnou komunikaci zajišťuje rodina protokolů TCP/IP. Pro identifikaci jednotlivých uzlů se používá IP adresa.

Nejdříve byl vytvořen protokol TCP a potom byl k němu připojen protokol IP. Každá síť se dá představit na referenčním modelu ISO/OSI. Mezinárodní organizace pro normalizaci (ISO) vytvořila tento model, využívající vrstev, už v roce 1979. Tento model je založen na vzájemné komunikaci, při které se odesílají a přijímají data, mezi dvěma systémy. Model ISO/OSI obsahuje aplikační, prezentační, relační, transportní, síťovou, linkovou a fyzickou vrstvu dle sestupného seřazení. Systém odesílající data využívá pro tento proces všechny vrstvy, od aplikační až po fyzickou. Takto odeslané informace může přijímající systém zachytit buď na nejspodnější vrstvě (fyzické) nebo na kterékoli vyšší vrstvě. Při získávání těchto údajů, v přijímajícím systému, se zapojují vždy postupně vyšší vrstvy (od fyzické až po aplikační). Rodina protokolů TCP/IP je založena na modelu ISO/OSI, ale rozdíl spočívá ve sloučení určitých vrstev kvůli neúplně vyhovujícímu, dříve zmíněnému modelu.

Název	Charakteristika	Příklad
Aplikační vrstva	S využitím protokolu HTTP umožňuje přenos dat aplikacím	HTTP(S)
Transportní vrstva	Zprostředkovává spojení s jinými zařízeními	TCP
Síťová vrstva	Sleduje a řídí přenos paketů	IP
Vrstva síťového rozhraní	Vysílá a přijímá data	Ethernet, token ring...

Tab. 2.1-1 Vrstvy TCP/IP

V tabulce výše, konkrétně u prvních tří vrstev, jsou uvedeny právě ty příklady protokolů, které se týkají zobrazování webových stránek klientům. K aplikační vrstvě je možné přidat jako příklad i HTML nebo XML, jelikož se provádí transformace datové struktury do požadovaného formátu podle standardu. V modelu ISO/OSI tuto úlohu neplnila aplikační vrstva, ale vrstva prezentační.[2]

2.2. Intranet

Intranet je privátní počítačová síť, která umožňuje v rámci organizace efektivní komunikaci, sdílení informací i spolupráci, a tím pomáhá zvyšovat výkon práce. Intranet se taky dá popsat jako organizační web, ke kterému mají přístup buď všechna počítačová zařízení v rámci organizace, anebo jen určité části či oddělení v téže organizaci. Z toho vyplývá, že by se intranet dal považovat za soukromou společnou síť sdílených informací a dokumentů. Při vytváření intranetu se může využít stávající sítě LAN i WAN, ale je chybné porovnávat tyto klasické sítě a intranet, protože intranet je založen na specifických technologiích standardů a protokolů. Konkrétně se jedná o protokoly TCP/IP a HTTP a potom o manipulační jazyk HTML.[3]

2.2.1. Rozdíly mezi Internetem a intranetem

Základním rozdílem mezi Internetem a intranetem je fakt, že i když oba používají, ať už částečně nebo úplně, webový server pro svoji funkcionalitu, je webový server intranetu připojen pouze k lokální síti dané organizace. Kromě toho může intranet využívat i poštovních nebo diskuzních serverů. Intranet, na rozdíl od Internetu, tedy může poskytovat libovolné Internetové standardy, které jsou ale ovšem přístupné pouze v rámci vnitřní infrastruktury instituce. Častokrát můžou lidé využívající tuto vnitřní síť využívat i vnější síť Internet, ale externí uživatelé, bez potřebného pověření, nemají přístup k datům ani vnitřní struktuře intranetu. Existují společnosti, které využívají dat v intranetu nejen pro vnitřní potřeby, ale taky tyto data zpřístupňují i osobám na Internetu, ale pouze při vymezených podmínkách.[4]

2.2.2. Výhody a nevýhody

Intranet je tvořen pěti hlavními funkcemi, které můžou být využívány buďto jednotlivě, nebo kombinací jednotlivých funkcí. Mezi tyto funkce řadíme:

- **E-mail** umožňující vzájemnou komunikaci mezi uživateli nebo skupinami uživatelů,
- **Adresáře** poskytující strukturu dat a informací,
- **Správu sítě** dovolující řízení intranetu,
- **Sdílení souborů** zprostředkující užívání dat a informací od jiných uživatelů sítě nebo skupiny uživatelů,
- **Hledání** jakýchkoli souborů, dat a informací.

Díky těmto funkcím intranetu se zvyšuje už dříve zmíněná produktivita práce, jelikož tyto funkce šetří čas a energii pracovníků. Na druhou stranu intranet nešetří pouze čas, ale může změnit způsob vykonávání práce a tím přinést mnoho výhod do rutinně zavedených institucí.[3]

2.2.3. Zabezpečení

Obecnou definicí pro zabezpečení jakékoli sítě je určit, kdo může přistupovat na síť, ochrana soukromých dat a taky jejich vzájemná integrita. Pro zajištění kvalitní bezpečnosti se nejčastěji využívá kombinace softwaru, hardwaru a administrativních strategií. Zabezpečování a ochrana sítě je stále trvající vyvíjející se proces, protože útoky a snahy prolomit ochranu se stále vyvíjí.[3]

a) Interní ochrana

Vnitřní zabezpečení informací je stejně důležité jako zabezpečení vnější. Narušená ochrana dat může vzniknout, jestliže data jsou načítána z Internetu, heslo k počítači či k interním aplikacím je na viditelném místě, není ošetřená bezpečnost dat při neodhlášení od počítače nebo odchodem a může vést ke ztrátě, zcizení, úniku mnohokrát citlivých informací. Ochranné opatření může být školení ohledně bezpečnosti pracovníků pracujících s intranetem a má být samozřejmostí ve všech organizacích. To samé platí i pro všechny ostatní uživatele dat, kteří mají přístup k těmto interním datům.

K dalším interním nebezpečím lze připočíst i zaměstnance, kteří už nepracují pro danou společnost a mají vůči ní nějaké nevyřešené problémy. Tito bývalí zaměstnanci můžou znát stará hesla k zařízením nebo aplikacím, a proto by opět měla být samozřejmostí aktualizace přístupových práv anebo například pravidelná změna hesel. Jako poslední příklad slouží lidé, kteří pracují v dané instituci a každý den mají přístup k změnám a mazání dat. Takováto rizika se eliminují například častým zálohováním dat na jiné datové zařízení a v případě, že by takovýto útok nastal, se provede aktualizace dat právě z této provedené zálohy.[3]

b) Externí ochrana

S externí ochranou přímo souvisí vnější nebezpečí. Do tohoto nebezpečí se započítává ohrožení za strany hackerů, ale taky ze strany vyhledávacích robotů u těch intranetů, které používají sdílení dat i s veřejnou sítí, protože vyhledávací roboty můžou archivovat neveřejná data.

U případu s hackery může jít o cílené útoky z důvodu krádeže dat a jejich následného prodeje, nebo jen o pocit činu nebo zábavy, kterým si dokazují, čeho všeho jsou se svými dovednostmi schopní. Pro velké společnosti je lepší si ještě před zavedením sítě najmout vyhlášeného hackera nebo hackery, kteří poukáží na slabiny dané sítě. Z oněch výsledků lze vytvořit zabezpečení takovéto sítě ještě pevnější a stabilnější. Pro zabezpečení sítě se používají nejběžněji firewally, ověření pravosti, např. autorizační sms apod., souměrné či nesouměrné zašifrování dat, ale taky přenosy těchto dat, nebo v neposlední řadě využívání zabezpečovacích certifikátů, eventuálně elektronických podpisů. [3]

2.2.4. Webové aplikace na intranetu

Intranet pro webovou službu obsahuje webový server, a proto aplikace na takovém serveru je úplně stejná jako na webových serverech na Internetu. Samozřejmě při programování takovéto aplikace se vyskytnou rozdíly v mnoha aspektech. Za prvé v zabezpečení, a to záleží na tom, jestli daná aplikace je přístupná z Internetu nebo pouze intranetu; za druhé v rozdělení podle toho, v jaké instituci a v jakém intranetu se nachází. Dalšími důležitými prvky v aplikaci jsou rychlost, přístupnost, integrita a spolehlivost.

Ještě před začátkem programování intranetové aplikace je nutno určit plán obsahu, se kterým se bude v této aplikaci pracovat. Je důležité si určit cíle vytvářených stránek a jak těchto cílů dosáhnout. Proto je důležité posoudit např. už dříve zavedené dokumenty a jejich strukturu, informační materiály, vazby mezi jednotlivými písemnostmi apod. Tyto kroky upřesňují, jaký plán vytvořit a čeho se při programování držet.

Pro webové aplikace pracující na intranetu, ale taky na Internetu, je vhodné co nejvíc tyto aplikace testovat, jak už z hlediska funkčnosti, tak bezpečnosti. Z hlediska funkčnosti je nejlepší možností nechat testovat aplikaci přímo zaměstnanci, kteří ji neprogramovali, a proto můžou zjistit jakékoli funkční nedostatky. Následně po tomto zjištění je dobré všechny požadavky a připomínky projít a případně opravit. U kvalitních intranetových aplikací je nutností zapojit stránku se zpětnou vazbou, případně kontaktem na technickou podporu.[4]

Při programování webových aplikací pracujících s elektronickými dokumenty pro veřejnou správu se nesmí zapomenout na povinnost, kterou je dodržování zákonů České republiky. Mezi tyto zákony řadíme např. zákon 101/2000 Sb. o ochraně osobních údajů a o změně některých zákonů, zákon 102/1971 Sb. o ochraně státního tajemství a zákon 227/2000 Sb. o elektronickém podpisu.[5]

2.3. Statické a dynamické webové stránky

Služba WWW (World Wide Web) jsou službou fungující na celosvětové počítačové sítí, Internetu, ale taky i na intranetu. Služba WWW je tvořena čtyřmi hlavními technologiemi:

- webovými stránkami (HTML, XHTML, CSS),
- URL,
- protokolem HTTP,
- principem klient/server.

Webová stránka je textový dokument, který je složen z hypertextového značkovacího jazyka (HTML), který umožňuje používat odkazy na jiné stránky, vkládat obrázky, popř. videa, a mnoho dalšího.

Statické webové stránky jsou stránky v čase neměnné, to znamená takové, které se po určitém časovém horizontu nijak nezmění s výjimkou, že by byl změněn jejich zdrojový kód. Statické webové stránky se charakterizují tím, že mají zanedbatelně malé požadavky na webový server. Mezi jejich výhody patří nejen jednoduchost kódu, ale taky nízká pořizovací cena a rychlost. Statické stránky jsou vhodné jako internetová vizitka nebo jednoduché prezentační weby.

Dynamické webové stránky, oproti stránkám statickým, umožňují měnit obsah stránek bez nutnosti zasahovat do zdrojového kódu. Pro množnost měnit webové stránky je nutný skriptovací jazyk, který neumožňuje pouze měnit, ale taky se podílí na tvoření webových stránek a tím přidává stránkám dynamiku. Skriptovací jazyky, které jsou tedy i programovací jazyky, se rozdělují na klientské a serverové skriptovací jazyky. Klientské skriptovací jazyky má na starost prohlížeč nebo taky klient. Tyto skripty umožňují využívat různé efekty pro zpřehlednění akcí na stránkách, zpříjemnění práce ve webové aplikaci atd. Mezi tyto jazyky počítáme např. JavaScript, JScript, VBScript apod. Serverové skriptovací jazyky na rozdíl od klientských jsou zajišťovány serverem. Pomocí serverových skriptovacích jazyků se programují dynamické webové aplikace poskytující práci s databází, e-maily, sdílením dat na stránkách atd. Mezi současné skriptovací jazyky se řadí čtyři základní. Jsou jimi ASP, PHP, JSP a ASP.NET.[11]

2.4. HTML a XHTML

HyperText Markup Language (HTML) je hypertextový značkovací jazyk. Jazyk HTML je nejčastěji používaným jazykem k tvorbě webových stránek. Byl vytvořen Timem

Bernersem-Leem a Robertem Caillau v roce 1989. Tento jazyk má mnoho verzí. Verze 2.0 vyšla v polovině roku 1994, verze 3.2 se objevila začátkem roku 1996, verze 4.0 byla vypuštěna do světa koncem roku 1997 a verze 4.01 vznikla koncem roku 1999.

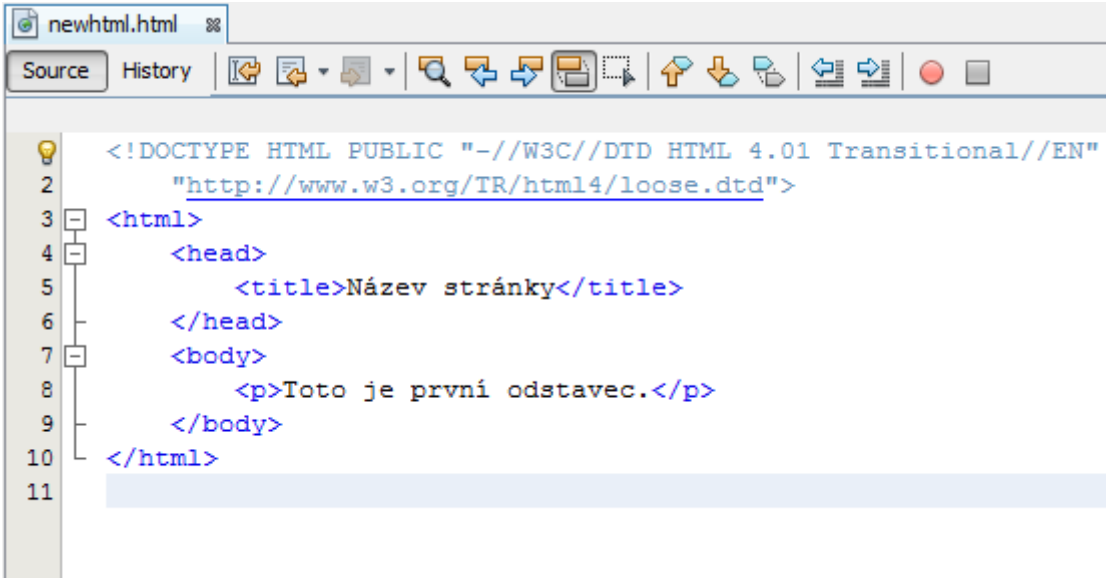
Jelikož je vytváření standardů vždy velmi obtížné, tak proto se od verze 3.2 vývoj standardů přesunul do rukou World Wide Web Consortium (W3C). Tato změna byla nutností, jelikož postupem času začali tvůrci prohlížečů nejen chápat jazyk odlišnými způsoby, ale taky se lišila podpora různých elementů jazyka v jednotlivých webových prohlížečích. Díky standardům konsorcia je tedy položen základ HTML jazyka, na kterém můžou stavět jak programátoři či vývojáři webových prohlížečů, tak i webdesignéři, kterým by tyto standarty měly zajišťovat jistotu, že jednotlivé povolené elementy jazyka fungují ve všech současných prohlížečích.[1]

Jak již bylo uvedeno výše, jazyk HTML využívá značky (angl. tag). Jednotlivé značky jsou příkazy uprostřed ostrých závorek. Některé značky můžou obsahovat své vlastní parametry, čímž jsou dané značky blíže specifikovány. Parametry značky se přiřazují názvům parametrů pomocí znaku rovnítka.

Tim Berners-Lee zpočátku využíval značek Standard Generalized Markup Language (SGML), ale postupem času začal používat jazyk pro definování značek, přesněji DocType Definition (DTD). Tento jazyk se vyznačuje tím, že určuje strukturu, tzn. v tomto případě značky, jejich vlastnosti, pořadí, parametry atd. Jazyk DTD není pro programování v jazyce HTML nutností, ale jde ho využít při vytváření a čtení XML jazyka, jelikož je na něm založen.

Každá webová stránka obsahuje základní strukturu skládající se z HTML značek. V této základní struktuře jsou všechny značky párové. Jako první se uvádí značka `<html>`. Tato značka je nadřízená všem ostatním značkám ve stránce a je ukončena jako všechny párové značky lomítkem a názvem. Jako další je hlavička se značkou `<head>`, která může obsahovat další podřízené značky jako název stránky `<title>`, přiřazené externí styly `<link>`, skripty `<script>` nebo značky `<meta>`, které popisují obsah stránek a další. Následuje značka `<body>`, která se označuje jako „tělo“ stránky a obsahuje všechno, co se na stránce zobrazí. Může obsahovat jakékoli texty, odkazy, obrázky, tabulky, animace apod. Je ještě velmi důležité zmínit, že před značku `<html>` se vkládá ještě údaj o využívané verzi (X)HTML, tzv. DOCTYPE. Významem DOCTYPE je, že je používán při validaci, ale taky že s různými variantami DOCTYPE se mění vzhled výsledných stránek. DOCTYPE slouží

k určení definice dokumentu (DTD). Rozlišují se tři základní verze DTD, a to transitional, strict a frameset. Na následujícím obrázku je možno vidět základní kostru jazyka HTML s přidaným názvem stránky a jedním odstavcem v programu NetBeans.[2]



Obr. 2-1 Základní struktura HTML

Postupným technologickým vývojem se na trh se zařízeními, využívajícími zobrazování webových stránek, začalo dostávat více a více společností s různými přístroji. Mezi tyto přístroje je možno považovat mobilní telefony, kapesní počítače (PDA), hlasové prohlížeče apod. Jenže tato zařízení měla různé potřeby a jazyk HTML je nebyl schopen naplňovat. V tomto pohledu zasáhlo konsorcium W3 tím, že vytvořilo nový jazyk XML (eXtensible Markup Language), což je rozšířitelný značkovací jazyk, který řeší nedostatky jazyka HTML, jako třeba zpracování, zobrazení a rozšiřitelnost.

Pomocí jazyka XML se vytváří dokumenty obsahující strukturovaná data. Jazyk XHTML (eXtensible HTML) byl vytvořen na základě jazyka XML stejně jako DTD. Aplikace založené na XML můžou využívat stejných pravidel, a proto je pro prohlížeč kvůli jednoduchosti mnohem jednodušší zobrazení XHTML, než HTML. Slouží pro tvorbu webových stránek, ale v tomto případě se kombinují vlastnosti jazyka HTML a vlastnosti XML, díky čemuž má XHTML i striktní předpisy. Těmito předpisy se myslí např. respektování velkých a malých písmen, dodržování uzavíraní značek (i nepárových), správné vnořování značek atd.[1]

Mezi další rozdíly XHTML a HTML patří možnost uvést pro webovou stránku jmenný prostor (namespace). Tento jmenný prostor se zapisuje v XHTML do vybrané

značky[1], a to znamená, že všechny její podřízené značky musí tento jmenný prostor dodržovat. Využívání jmenného prostoru není pro stránky nutností, ale pokud je záměrem držet se specifikace, pak by se jmenný prostor využívat měl, jelikož specifikace XML je vyžaduje.

XHTML jde taky rozdělit podle významových skupin. V rozdělení podle DOCTYPE, patří verze „Strict“, která striktně, jak už z názvu vyplývá, omezuje na používání nových značek a na staré značky nebere žádný zřetel. Druhá verze je „Transitional“, která zachovává zpětnou kompatibilitu s HTML 4.01, a poslední je „Frameset“, která je stejná jako „Transitional“, ale umožňuje využívání značek FRAMESET. V rozdělení podle verzí se XHTML rozděluje na 1.0 a 1.1. Pro verzi 1.0 platí další rozdělení podle DOCTYPE, ale pro verzi 1.1 už ne. Je tak proto, že verze 1.1 je založená na verzi 1.0 Strict a z tohoto důvodu se například o rámcích vůbec neuvažuje. Výhodou XHTML 1.1 je třeba možnost rozšíření o různé moduly. Takovým modulem můžou být právě rámce.[2]

Novinkou ve formátu HTML je jeho pátá verze, která obsahuje staré i nové potřeby webů. Mnoho věcí v HTML 5 jsou stejné jako v původní HTML 4 nebo XHTML 1, ale zároveň přináší nové možnosti. Mezi tyto možnosti je možno započítat sekce (section), články a komentáře (article), přílohy nemající vliv na hlavní obsah (figure) a mnoho dalších. Jednotlivé prohlížeče postupně implementují čím dál víc prvků z HTML 5, ale části tohoto jazyka jsou stále ve fázi dokončování, proto se nejedná ještě o konečnou verzi. I když jazyk HTML 5 zatím není plně podporován v prohlížečích, už nyní je možnost využívat jeho výhod, protože z větší části již podporován je.[6]

2.5. CSS – Kaskádové styly

Kaskádové styly (angl. Cascading Style Sheets) byly vytvořeny spolu s HTML verze 4.0 v roce 1996 s názvem „CSS 1“. Postupem času se objevila v roce 1998 verze 2.0, která přinesla novinky jako relativní a absolutní pozicování, vlastnost z-index určující pořadí zobrazování prvku na stránce, obousměrný text apod. V současné době je využívána verze 2.1.[2] Ve vývoji se nachází verze CSS 3, která je rozdělená na dílčí moduly. Určité moduly jako selektory 3. verze, jmenné prostory a další už teď jsou určeny za finální (Recommendation). Další jsou ve fázi kandidátů, u kterých se finální verze ještě přepracovává, a u ostatních se čeká na další vývoj.[6]

[1] Příklad vložení jmenného prostoru do značky html může být: `<html xmlns="http://www.w3.org/1999/xhtml">`

Primárním účelem CSS je oddělit informační stránku webu od vzhledové stránky. V případě použití HTML s kaskádovými styly se značkovací jazyk stará o věcnou stránku a úloha vzhledu je dána na starost CSS. Toto rozdělení má výhodu v tom, že se už nemusí provádět formátování vzhledu prostřednictvím HTML značek, ale právě díky kaskádovým stylům. Toto značně zpřehledňuje kód u obou dvou způsobů.[2]

Díky CSS je možnost měnit vzhled webu na úrovni místí nebo globální. Umožňuje nastavovat např. velikost, styl a typ písma, jeho barvu, ohraničení, zarovnání. Poskytuje udávat rozměry prvků, jejich pořadí a mnoho dalšího. Další výhoda kaskádových stylů je, že při jejich použití na globální úrovni, se tyto styly kaskádově rozšíří do všech stránek bez nutnosti zásahu v jednotlivých kódech HTML.

Kaskádové styly jsou tvořeny tak, že v kódu CSS jsou přiřazeny různým HTML značkám libovolné styly. V druhém případě je možno vytvořit svůj vlastní styl a následně ho nějaké HTML značce přiřadit. Přiřazení vlastních stylů značkám se provádí prostřednictvím **tříd** (class) a **identifikátorů** (id). Platí konvence, že třídy jsou používány v případech, kdy se jeden styl přiřazuje více značkám. Naopak platí konvence pro identifikátory, že jsou použity tehdy, když jsou styly u značek jedinečné. Oba selektory se zapisují do jazyka HTML způsobem, že jsou připsány k jednotlivým značkám, ke kterým chce programátor připojit styl, atributy buď „class“ nebo „id“ a přiřadí se k nim název stylu pomocí rovnítka, např. `<p class="is">`.[1]

2.6. JavaScript

Je to programovací jazyk, který patří mezi klientské skriptovací jazyky. Tento jazyk byl vytvořen společností Netscape v roce 1995, a jak už bylo dříve zmíněno, jeho chod zajišťuje klientský prohlížeč, přičemž kód JavaScriptu se vykonává až po jejím načtení. Díky této vlastnosti z bezpečnostních důvodů neexistují funkce pro správu souborů. Na druhou stranu je možno využít jazyk k zatraktivnění aplikace nebo webové stránky díky různým interakcím s událostmi, ale i díky podmínkám, proměnným a cyklům. Jazyk JavaScript se nejčastěji na webových stránkách používá k:

- záměně obrázků při přejetí myší,
- práci s dialogovými okny,
- práci s datem a časem a jejich následném zobrazení (kalendář, hodiny),
- vytvoření víceúrovňových a roletkových nabídek,

- kontrolu správnosti údajů zadaných uživatelem do formuláře a jejich následnou opravu nebo nápovědu ještě před odesláním stránky na server.

JavaScriptový kód se na stránkách zapisuje do párové značky „script“. Dříve se používala kombinace atributu „type“, který určuje MIME typ skriptu s atributem „language“, který popisuje jazyk skriptovacího jazyka. Nyní se ale, s použitím XHTML, používá pouze „type“. Příklad interního zápisu může být `<script type="text/javascript"></script>`. Pokud developer chce, aby se kód provedl ještě před načtením obsahu, pak umístí tento zápis do hlavičky stránky. Pokud nevyžaduje, aby byl skript proveden před načtením stránky, může ho umístit kdekoli do těla stránky. Díky tomuto zápisu prohlížeč pozná, že obsah nemá zobrazovat, ale vykonávat. Skriptový externí soubor jde připojit ke stránce stejným způsobem jako interní skript, jen s tím rozdílem, že se přidá atribut „src“ a relativní cesta k externímu skriptu. V ostatních vlastnostech, včetně začleňování do těla i do hlavičky stránky, zůstává externí skript shodný s interním.[7]

Příkazy JavaScriptu lze rozdělit na dvě hlavní skupiny skriptů, a to:

- ty, které se vykonají pouze jedenkrát, nejčastěji po načtení dokumentu a
- ty, které reagují na jisté události zapříčiněné například najetím myši na daný objekt, stisknutím určitého tlačítka atd.

Jedním z nejdůležitějších funkcí jazyka je základní výpis textu, ať už formátovaného nebo ne, na stránku. Tento výpis se provádí příkazem `document.write()`.[8]

Knihovna jQuery je knihovna založena na jazyce JavaScript a je navržena pro ulehčení práce při vytváření ať už JavaScriptových aplikací nebo jakékoli tvorbě animací či zjednodušení. JQuery umožňuje při psaní méně kódu, dosáhnout stejného výsledku, jakoby byl použit čistý kód JavaScript a proto práci velmi zrychluje a zefektivňuje.

Poprvé byla knihovna jQuery představena v roce 2006 přes Johna Resiga, který ji zároveň vytvořil a v současnosti je jejím hlavním vývojářem. Tato knihovna je šířena pod licencemi GPL a MIT, díky kterým je jQuery volně použitelná, ale taky umožňují komukoliv zapojení se do jejího vývoje.

Knihovna jQuery se zaměřuje na využití komunikace jazyka JavaScript s jazykem HTML. Díky tomu, že umožňuje používat nejrůznější zabudované funkce pro animace, události, mnoho komponent atd. její uživatel skoro nemusí využívat technologie Flash, u které je známo, že má větší datový objem a je méně dostupná pro internetové vyhledávače.

Knihovna vyniká svým využitím i u formulářových prvků, kde může značně ulehčovat orientaci nad tím, jaké požadavky jsou kladeny při vyplňování právě jednotlivých formulářových prvků a taky zobrazovat nápovědu. Umožňuje i např. graficky efektní upozornění uživatele formuláře v případě, že formulář nebyl vyplněn správně.

Mezi další výhody této knihovny patří i významný počet zásuvných modulů, mezi něž patří „Carousel“, „Date picker“ atd. Programátoři si můžou naprogramovat své vlastní zásuvné moduly a následně je distribuovat. Jako velkou výhodu lze považovat i ve výběru elementů v rámci modelu DOM, které jsou vybírány prostřednictvím výběrového jádra Sizzle. Mimo výběr elementů patří mezi silné stránky jQuery i jejich modifikace. Díky této modifikaci můžou být kaskádové styly elementů jednoduše změněny. Jako další klad se dá brát i skutečnost, že všechny části knihovny jQuery jsou podporovány ve všech současných webových prohlížečích. Jako významný klad lze považovat i už dříve zmíněné události. Mezi tyto události je možno řadit např. onMouseOver (při přesunutí myši na daný objekt), onFocus (při vybrání určitého objektu) apod. Události značně zpestřují rozsah knihovny a tím přispívají k úplnosti.[9]

Knihovnu jQuery lze využít i k spolupráci s technologií AJAX. V knihovně je možné nalézt mnoho metod pro komunikaci se serverem, tj. vytváření serverových požadavků, načítání externího obsahu, práce s událostmi požadavků atd. V případě, že by se ve webové aplikaci měl vytvořit pouze jeden serverový požadavek, pak by moc velký rozdíl při použití jQuery nenastal. Ale v případě, že je nutno vytvořit požadavků víc, je přínos jQuery neocenitelný.[10]

2.7. PHP

PHP (Hypertext Preprocesor) je programovací jazyk, který řadíme mezi serverové skriptové jazyky, jelikož je zpracováván a zajišťován serverem. PHP je velmi populární díky své jednoduchosti a funkčnosti na různých operačních systémech, ale hlavně webových serverech. Tento jazyk poskytuje silný nástroj pro tvorbu dynamických webových stránek s mnoha možnostmi.[12]

2.7.1. Historie

První verze PHP, zvaná PHP/FI byla vytvořena v roce 1995 Rasmusem Lerdorfem na základě jazyka Perl. Jelikož se tato sada skriptů stala postupem času velmi žádanou, tak se v roce 1997 objevila druhá verze tohoto jazyka. Následně chtěli Andi Gutmans a Zeev Suranski využít PHP/FI verze 2 na programování komerčních aplikací, ale tato verze nebyla

dostačující, a proto ji kompletně předělali a výsledek označili jako PHP 3. První elementy objektového programování se objevily už v této verzi. Díky tomu, že PHP třetí verze nestačilo pro složitější a komplikovanější aplikace, bylo nutno provést změnu PHP jádra. Toto jádro bylo označováno zkratkou ZEND, která vznikla z kombinace křestních jmen vývojářů zmíněných dříve u třetí verze. Tato změna začala v zimě roku 1998 a následně byla v roce 2000 dokončena a označena jako verze 4.[11] Tato verze obsahovala už nějaké schopnosti objektově-orientovaného programování, jako například volání statických metod, ale byl to natolik nezdařilý pokus, že mnoho lidí nabylo dojmu, že PHP jako skriptovací jazyk není pro OO programování vhodný.[12] Následně v roce 2004 byla vyvinuta verze PHP 5, která přinesla mnoho vylepšení do objektově-orientovaného programování.[11] Mezi tyto změny v OO programování lze považovat např. abstraktní rozhraní, viditelnosti tříd, konstruktory, ale i změny v ošetření výjimek a mnoho dalších. Proto se i názory mnoha lidí na objektově-orientované PHP změnily. V současné době se používá PHP verze 5.4 a pracuje na verzi 6.0. Ve verzi 5.4 není tento jazyk plně objektově-orientovaný, ale jen částečně, jelikož některé základní datové typy jako „string“, „number“ atd. nejsou objekty, a některé části knihovny jsou stále považovány za funkce. To ale neznamená, že PHP není OO jazyk.[12]

2.7.2. Požadavky provozu

Pro práci s PHP skripty je zapotřebí server. Konkrétně se jedná o webový server, který podporuje právě tento jazyk. Na druhou stranu lze použít kterýkoli webový server s podporou rozhraní CGI a podporu PHP skriptu tam nainstalovat. Je možno využívat za určitý poplatek webový server přes Internet, ale pro vývoj webové aplikace to není nejlepší řešení. Mezi nevýhody této varianty jde přičíst neustálé přenášení souborů a nutnost připojení k Internetu. Při vývoji aplikací se nejčastěji volí webový server zprovoznitelný na lokálním počítači např. Apache. Při zavádění tohoto serveru je často k dispozici i databáze, např. MySQL, a databázový správce, např. phpMyAdmin. Mezi velké výhody tohoto řešení lze počítat rychlost a neomezenost spojení, nezávislost, ale i možnost vytvoření „virtuálních hostů“, kteří umožňují na lokálním počítači nastavit více domén např. pro jednotlivé webové aplikace. Mezi slabé stránky této varianty lze započítat např. testování transakčních metod z důvodu složitějšího nastavování konfiguračních souborů pro více počítačů v místní počítačové síti.[11]

2.7.3. Možnosti využití

Pro mnoho lidí představuje jazyk PHP nástroj pro vytvoření rozhraní databází, manipulaci s formulářovými informacemi, vytváření dynamicky generovaných webových

stránek apod. Tento jazyk ale dokáže rovněž vytvářet a upravovat Flash, obrazové a PDF soubory, komunikovat s protokolem LDAP, ověřovat uživatele na základě prostého databázového souboru, ale i Active Directory, komunikovat s velkou škálou protokolů jako IMAP, POP3, DNS a další. Mimo jiné dokáže spolupracovat s mnoha databázovými technologiemi, jako např. IBM DB2, MS SQL, MySQL, Oracle, PostgreSQL apod.[13]

2.7.4. Objektově-orientovaný jazyk

Objektově-orientovaný jazyk, znamená velkou změnu pro procedurálně orientovaného programátora. Objekty vnáší do programování možnosti lepšího zachycování objektů reálného světa, procesů a nápadů, se kterými přijde daná aplikace do styku. OOP umožňuje navrhovat aplikace vzájemným vztahem objektů, které mají své vlastnosti a činnosti s nimi spojené. Dalšími výhodami můžou být znovu použitelnost tříd, zřetelná struktura dokumentace, která přinese výsledky při programování ve skupině apod.

Do primárního konceptu objektově-orientovaného programování se řadí:

- Třída – vzor pro objekt, který kódově obsahuje vlastnosti a metody,
- Objekt – průběžná instance třídy, která zahrnuje všechna příslušná data a informace potřebné pro funkčnost aplikace,
- Dědičnost – schopnost vymezit třídu, která dědí vlastnosti a schopnosti ze své nadtřídy, kde může definovat navíc své vlastnosti,
- Polymorfismus – umožňuje definovat třídu jako člena více než jedné skupiny tříd,
- Rozhraní – způsob určení, že objekt je schopen provést nějakou činnost, aniž by se tato činnost přímo definovala,
- Zapouzdření – vlastnost objektu, která chrání v přístupu k vnitřním informacím.[15]

2.8. MySQL

Je to relační databázový systém, vytvořený komerční společností, který je přijat jako systém pro webové aplikace. MySQL při srovnání s ostatními databázovými systémy je podstatně jednodušší, nemá vysoké nároky na zdroje a u určitých operací umožňuje zvýšení výkonu, tj. rychlosti. Tím, že je systém relační, umožňuje spojení několika tabulek podle stupňů vzájemného vztahu.[11] E. F. Codd z IBM v roce 1970 definoval relační databáze jako sbírku informací seřazených v tabulkách, ve kterých se dají přidávat, modifikovat, likvidovat

nebo je načítat. Jsou-li data uspořádána efektivně, lze to provádět bez změny uspořádání tabulek.[14]

MySQL jako systém nabývá architekturu klient/server. Jako server je brán právě MySQL server a jako klient můžou být programy komunikující s daným serverem (webové aplikace atd.). Mezi tyto klienty lze řadit i skript PHP. Z toho vyplývá, že PHP může samo o sobě obsahovat příkazy jazyka SQL a na druhou stranu může s výsledky ze serveru dále pracovat. Tento server zpracovává velké množství dotazů ve strukturovaném dotazovacím jazyce (taky známým jako SQL).[11]

2.8.1. Nástroje

Pro používání databázového systému MySQL je potřebný MySQL server. Tento server může být buď oficiální volně přístupný server na všechny operační systémy, nebo kterýkoli jiný program, který bude tento server obsahovat, jako např. WAMP, XAMPP, PHP webový server atd. Tyto programy už buď samy zajistí běh databázového serveru, nebo se ho díky nim dá ovládat.

Pro správu databáze na databázovém serveru se nejčastěji používá nástroj phpMyAdmin. Tento prostředek umožňuje vykonávat všechny základní činnosti při práci s databází. Tato nadstavba se používá přes internetový prohlížeč, a proto je nejčastěji používaná u webhostingů. Mimo to existuje mnoho konkurenčních nástrojů pro správu databázového serveru. Alternativním způsobem pro správu databáze je i jednoduchý příkazový řádek, který je ve své funkcionalitě omezen právě a pouze funkcemi databázového serveru MySQL.[11]

2.8.2. Struktura databáze

MySQL lze teoreticky brát jako relační systém řízení báze dat (RSŘBD). Je schopen ukládat informace do, obecně řečeno, přehledných objektů, kterými jsou často tabulky, ale můžou jimi být i pohledy, procedury atd. Každý objekt, ze sémantického hlediska, má své charakteristiky. Tyto charakteristiky se u relačního modelu stávají sloupci v daných tabulkách. Sloupce můžou mít rozlišné typy, jako např. text, čísla, kalendářní data atp. Informačně a typově stejné sloupce z dvou tabulek se dá relačně spojit a tohoto spojení využívat.[14]

2.8.3. Typy úložišť

Server MySQL je na databázových úložištích funkčně velmi závislý, jelikož jsou zodpovědná za ukládání a načítání dat. MySQL poskytuje pro programátory a správce databází mnoho možností. Mezi tyto mechanizmy patří MyISAM, InnoDB, MERGE, MEMORY a další. Mezi dvě nejpoužívanější varianty je možno řadit MyISAM a InnoDB.

Úložiště InnoDB přináší lepší možnosti pro víceuživatelský přístup k databázi, protože poskytuje uzamykání na úrovni řádků a ne celých tabulek. Volí se v případě, že se provádí závažné transakce, např. při platebních, rezervačních systémech, to znamená takových, že se požaduje změna v datech najednou. Tato varianta je vhodná i pro velké objemy dat, při důležitosti integrity dat nebo při konfiguraci dat a souborů protokolů. Nevýhodou je, že InnoDB je, při obecnějším porovnání, pomalejší než MyISAM a zabírá více diskového prostoru.

MyISAM je nástupce uložiště ISAM, který je původní produkt MySQL. V některých případech je toto úložiště nastaveno jako výchozí. Jeho hlavní výhodou je velká rychlost zpracování operací, ale na úkor záruky integrity a správy transakcí. Při vkládání stejných informací do tabulky při použití jak MyISAM, tak InnoDB, se operace v úložišti MyISAM provedly o 20% rychleji než v případě opačném. Na druhou stranu lze úložiště InnoDB pro netransakční operace optimalizovat na větší výkon. MyISAM dovoluje propracovanější volby indexování, včetně indexů sloupců BLOB a TEXT. [16]

2.8.4. Licencování

Při používání databázového systému MySQL na webových stránkách je tato možnost celkově zdarma. Je tomu tak proto, že při využívání tohoto systému na webhostinzích se uvažuje o jeho licenci GNU GPL, která umožňuje jeho používání, ale pouze v případech, že jsou zdrojové soubory vytvořeného programu k dispozici a není potřeba záruky správnosti této technologie. Může být určena záruka, ale pouze tvůrcem programu. Tato teorie platí i pro jakékoli jiné používání, např. komerční, ale musí být pravidla GNU GPL dodrženy.[2] V případě nedodržení, ať už z jakýchkoli důvodů, licence GNU GPL, je nutno zaplatit tvůrcům licenci.[11]

[2] Licence GNU GPL je volně dostupná na adrese: http://www.gnu.org/licenses/gpl.html

2.9. Zend Framework

Je to nástroj pro zrychlení, ulehčení práce programátora v jazyce PHP, díky už dřív vytvořeným komponentám, architektuře a dalším možnostem. Patří do skupiny OpenSource PHP frameworků a je podporován firmou Zend Technologies, která se podílí mj. i na vývoji jazyka PHP. Mnoho společností je partnerem právě tohoto projektu a případně poskytují zdrojové soubory, jak pracovat s jejich aplikacemi. Jsou to firmy jako IBM, Microsoft, Google, Adobe atd. Zend Framework má velké zázemí v komunitě vývojářů, kteří se podílejí na vývoji jednotlivých komponent, ale taky upozorní na chyby, se kterými se setkali. Velkou výhodou Zend Frameworku oproti ostatní frameworkům je fakt, že každé vydání nové verze je kompatibilní s funkcemi a komponentami z verzí starších, takže se nemůže stát, že po updatu na novou verzi nám už námi využívané komponenty začnou hlásit chyby. Další silná stránka je velmi podrobná dokumentace, u které se její sepsání provádí vždy před vypuštěním nějaké nové komponenty nebo nové verze. V současné době se vyvíjí ZF verze 2.1.1, ale stabilní je pořád verze 1.12.1. Druhá verze přináší mnoho vylepšení s příchodem PHP 5.3.3, jako jmenné prostory, upravenou architekturu aplikace a mnoho dalších. A i přesto, že vývojáři druhé verze doporučují přechod právě na ni, tak je ještě ve fázi raného vývoje a neobsahuje velmi užitečné komponenty, jako např. Zend Tool ad., a proto je stále udržována i verze první. Novinky, které přinesla druhá verze, byly nakonec použity částečně i k verzi 1.12, a jelikož je podpora první verze, např. návody, publikace apod. nesrovnatelně větší než u verze druhé, bude se popis Zend Frameworku v této kapitole zaměřovat na funkcionalitu verze 1. Navíc převod aplikace z verze první na druhou bude možný. Přesto je ale žádoucí, aby si programátoři už zvykali na verzi novou a učili se ji[3].

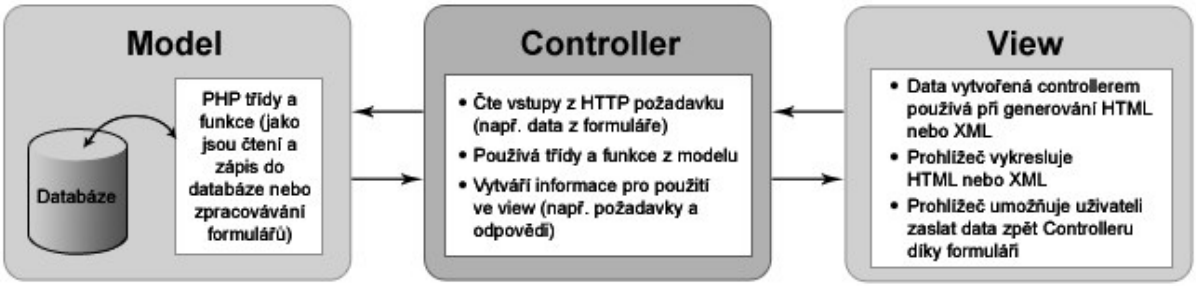
Požadavky na provoz Zend Frameworku jsou samozřejmostí, když se jedná o framework nad jazykem PHP. Mezi tyto požadavky se počítá webový server s databází a PHP verze 5.2.4 a vyšší. Jako další požadavek je funkcionalita `mod_rewrite`, který umožňuje přepis URL adres na přehlednější typ. U PHP rozšíření se požadují `ctype`, `pcre`, `Reflection`, `session` a `SPL`. Ve většině případů jsou tyto rozšíření už aktivované. V případě využívání PDO adaptéru pro práci s MySQL je třeba aktivovat ještě rozšíření `pdo` a `pdo_mysql`.[17]

[3] Více informací na oficiální adrese ZF: http://framework.zend.com/

2.9.1. Architektura

Zend Framework pracuje na modelovém vzoru MVC (Model – View – Controller, volně přeloženo jako Model – Pohled – Řadič). Tento vzor je často využívaný u OO programování webových aplikací. Model rozděluje prezentační stránku a logickou stránku aplikace. Do MVC modelu patří:

- **Model –** představuje aplikační logiku, do které se řadí spolupráce s databází, posílání emailových zpráv, zpracovávání peněžních transakcí atd.,
- **View** – prezentuje uživatelské rozhraní, které spolupracuje ve webových aplikacích nejčastěji s kódem HTML,
- **Controller** – spojující View a Model, umožňuje řídit procesy, události a požadavky v aplikaci.[18]



Obr. 2-2 Architektura Model – View – Controller [18]

Tento framework s sebou přináší svoji vlastní adresářovou strukturu. Na nejvyšší adresářové úrovni se nachází adresáře `appliaction`, `data`, `docs`, `library`, `public`, `scripts`, `temp` a `tests`. Je vhodné zmínit významy adresářů `appliaction`, `docs`, `library` a `public`. Velmi důležitý je adresář `application`, který obsahuje adresáře a soubory, mezi které patří:

- `configs/` – tady jsou uschovány konfigurační soubory celé aplikace,
- `controllers/` – obsahují všechny řadiče aplikace a jejich akce,
- `forms/` – zahrnuje veškeré soubory (třídy) zastupující formuláře,
- `layouts/` – tento adresář slouží k uložení layoutu,
- `models/` – je vhodný pro, jak už název vypovídá, modely, nejčastěji databázové,

- `modules/` – adresář za účelem obsažení modulů aplikace, které pomáhají v přehlednosti a funkčnosti komplexní aplikace skládající se např. z internetového obchodu, fóra a blogu,
- `services/` – složka zahrnující webové služby,
- `views/` – adresář určený pro soubory pohledů a šablon, kde pohledy jsou ve formátu PHTML,
- `Bootstrap.php` – původně soubor pro konfiguraci ZF, ale s nástupem verze 1.8 se konfigurace přesunula právě tam, a proto v současné době může Bootstrap zůstat prázdný nebo obsahovat inicializační metody.

Adresář `docs` je vhodný pro uložení dokumentace k dané aplikaci. `Library` slouží k uschování Zend Frameworku a vlastní rozšíření a adresář `public` obsahuje soubory, které jsou míněny jako veřejně přístupné přes server. Je tam index celé aplikace, ale může obsahovat i obrázky, kaskádové styly, JavaScript apod.[19]

2.9.2. Průběh požadavku

Velmi důležitým procesem v Zend Frameworku je průběh požadavku. Tento průběh probíhá skrze mnoho tříd a je zachycen v níže uvedeném postupu.

a) Zavedení aplikace

Inicializace začíná zavoláním metody `dispatch`() ze třídy `Zend_Controller_Front` buď ze souboru `index.php` nebo z komponenty `Zend_Application`. Tato třída následně zavolá zásuvné moduly, pokud jsou nějaké vytvořené, pomocí Plugin Brokeru. Potom se inicializují zásuvné moduly, které můžou být třídami, které obsahují metody používané na více místech v aplikaci díky Action Brokeru. Tuto akci lze zamezit v případě, že se Action Helpery v aplikaci nevyžadují. Nato se volá objekt `Zend_Controller_Request_Http` pro zaregistrování všech proměnných získaných z prohlížeče, jako např. $_GET, $_POST atd., a uloží je do front controlleru a Plugin Brokeru. Stejně tak se volá i objekt `Zend_Controller_Response_Http`, který obsahuje proměnné pro zaslání zpět prohlížeči i s obsahem stránky, a zaregistruje se rovněž do front controlleru a Plugin Brokeru. Nakonec zavádění aplikace se inicializuje směrovač `Zend_Controller_Router_Rewrite`, jež stanoví na základě request objektu, který modul, action controller a action budou požadovány. To vše i s parametry dokáže tento směrovač poznat z pěkných URL adres a předá požadované informace dispečeru, který daný action controller zavolá.[17]

b) Zpracování

Při požadavku přechodu na jinou stránku Front controller předá request a response objekt komponentě `Zend_Controller_Dispatcher_Standart`. Tato komponenta, jak už bylo zmíněno dříve, zavolá příslušný action controller i action, a ještě předtím mu předá request a response objekt. Pokud byla akce vykonána, je předáno slovo opět dispečeru, který ověří, zda-li má být zavolána i jiná akce. Pokud ano, opakuje se volání této akce z příslušného action controlleru a postup se opakuje, dokud nejsou akce vykonány. Po vykonání akcí je zavolán front controller, která předá response objektu všechny hlavičky i obsah stránky pro odeslání na vykreslení. Tímto proces zpracování požadavku končí.[17]

2.9.3. Možnosti

Framework není nutno používat celý. Je možno využít pouze část a tu danou část nahrát na server. Stejně je to i s využitím jednotlivých komponent. Zend Framework obsahuje nepřeberné množství komponent. Je samozřejmostí, že v tomto frameworku jsou zabudované komponenty pro automatické načítání tříd podle potřeby, logování událostí, filtrování a ověřování vstupů atd. Mezi často využívané komponenty patří komponenty pro autentizaci, autorizaci a správu uživatelů. K dalším vymoženostem lze přičíst nahrání souborů a zabezpečení díky CAPTCHA ve formulářích. Usnadněná je lokalizace a vyhledávání ve webové aplikaci. Mezi výhody se dá počítat tvorba PDF dokumentů a komponenty pro komunikaci nebo využívání webových služeb, jako aplikace od Google, YouTube nebo jakékoli jiné služby.[17]

2.10. Vývojové prostředí NetBeans IDE

Open-source program NetBeans IDE je určen pro tvorbu aplikací jak desktopových, tak i webových a dalších, založených na jazyce Java, ale umožňuje současně kontrolovat syntaxi, zvýrazňovat proměnné, automaticky tvořit závorky, uvozovky a další u jazyků C/C++, PHP, JavaScript apod., a tím snižuje vynaložený čas programátora na práci. Díky tomu, že je lehce rozšiřitelný, dokáže implementovat jak cizí knihovny a následně jejich části nabízet při programování, tak i celé programovací jazyky. Mimo tyto výhody umožňuje efektivní přístup k souborům a disponuje pohodlnou nápovědou. Nevýhody tohoto programu jsou individuální podle toho, kterou část programu se využívá, ale obecně se mezi nevýhody počítá značné využívání operační paměti. Tuto záležitost lze ale vyřešit pomocí nastavení minimální a maximální hodnoty používané paměti v konfiguračním souboru NetBeans.[20]

3. Analýza současného stavu správy majetku

V první části kapitoly je analyzována činnost pracovníků majetkoprávního odboru. Jejich povinnosti, současný systém zpracování a hledání informací ohledně majetkových smluv a usnesení apod. V druhé části této kapitoly jsou uvedeny požadavky zaměstnanců na funkce a činnost aplikace. Kromě těchto požadavků lze nalézt i možnosti ze strany oddělení informatiky vzhledem ke zvolené technologii, na které je výsledný program založený, a jaké technologie jsou na Městském úřadu dostupné.

3.1. Analýza činností majetkoprávního odboru

K náplni majetkoprávního odboru na MěÚ v Českém Těšíně patří komplexní zajišťování správy majetku obce, tj. jeho nabývání, pronájem, zastavování, prodej a jiné formy disponování s tímto majetkem, včetně jeho evidence a kontroly.

V posledních letech šlo konkrétně o prodeje pozemků, budov, vč. souvisejících pozemků, věcná břemena a pronájmy pozemků. Dále nabývání majetku skrze dary a bezúplatné převody nemovitostí do majetku města. Následně výkupy pozemků, souhlasné prohlášení, dohody o souhlasu realizací staveb, zajišťování zápisů svěřeného nemovitého majetku obce záznamem v katastru nemovitostí na listy vlastnictví ve prospěch příspěvkových organizací města.

Pokud zaměstnanci majetkoprávního odboru potřebují ke své práci nahlédnout do již uzavřeného spisu, který se nachází v archivu v budově městského úřadu, musí si pro spis do archívu dojít, vyhledat patřičné dokumenty a znovu spis na své místo do archivu uložit, což je namáhavé a časově náročné.

Časově mnohem náročnější je, pokud už je spis odeslán do okresního archívu – pak si musí potřebné dokumenty písemně vyžádat a čekat, dokud mu patřičné dokumenty nebudou poskytnuty.

Mezi činnosti vedoucí majetkoprávního odboru patří i mj. průběžné kontroly podle sestavených plánů. Prověřují se věcné a obsahové správnosti uzavíraných smluv, včetně kontroly na ně navazujících úkonů (úhrady plateb, přiznání k dani z nemovitostí, provádění změn v účetní a operativní evidenci, plnění smluvních podmínek) a dodržování lhůt pro jednotlivá plnění. Měsíčně jsou prováděny kontroly stavu operativní evidence majetku na stavy účetní. Dále jsou měsíčně prováděny kontroly ohledně plnění a čerpání rozpočtu, a podle potřeby lze provádět rozpočtové úpravy a opatření.

3.2. Analýza požadavků a možností

Jak už bylo zmíněno na začátku kapitoly, je důležité, aby se nová aplikace řídila podle požadavků zaměstnanců, kteří jsou jejími uživateli, ale na druhé straně vycházela z možností oddělení informatiky, které tuto aplikaci zpřístupňuje právě zaměstnancům. Tyto požadavky vychází z jejich činností a měly by jim práci usnadnit.

Mezi požadavky zaměstnanců patří:

- evidence všech typů smluv a možnost jejich nahrání a získání,
- evidence jednotlivých usnesení ke smlouvám a i jejich možnosti nahrání a získání,
- možnost soubory smluv a usnesení nahrát jako přílohy (samostatné soubory ve formě naskenovaných dokumentů či elektronických dokumentů), nebo skrze vložení textů smluv, eventuálně usnesení a následné převedení textu do formy PDF dokumentů,
- funkce vytvoření uživatele a jeho schopnost přihlásit se do systému, následné umožnění změny osobních údajů i hesla, pouze v případě, že uživatel zná své staré heslo, jinak bez znalostí starého hesla může změnit heslo uživateli pouze správce,
- s uživateli spojené i pravomoci, které umožňují změnu a smazání pouze těch záznamů, které byly daným uživatelem vytvořené s výjimkou záznamů fyzických a právnických osob,
- evidence spisů, které jsou tvořeny usneseními a smlouvami,
- evidence nemovitostí a osob a možnost následného dohledání smlouvy, usnesení či spisu podle konkrétních údajů těchto dvou entit,
- vyhledávání a řazení záznamů v každé evidenci,
- vytváření úkolů správcem pro jednotlivé uživatele, kdy pouze správce může určit, zda byl daný úkol splněný nebo ne, uživatelé nemohou změnit ani smazat detaily úkolu vytvořeného správcem, jedině s výjimkou sobě vytvořených úkolů,
- úplná kontrola nad aplikací skrze účet správce,
- konzistence a rychlost programu,
- bezplatné nabytí aplikace.

Oddělení informatiky disponuje, vzhledem k zprovoznění a využívání aplikace, koncovými počítači s výpočetním výkonem 2,93 GHz (dvoujádrový procesor), 2 GB operační paměti, LCD monitory s rozlišením 1680x1050 obr. b., webovým serverem, licencemi na základní balíček Microsoft Office 2002 (konkrétně Word, Excel a PowerPoint), webovými prohlížeči Internet Explorer 8 i Mozilla Firefox verze 19, programem Adobe Reader pro zobrazování obsahu PDF dokumentů atd. Toto byly pouze vybrané možnosti oddělení týkající se problému řešeného v této práci.

Výkon a paměťová dispozice pracovních stanic jsou velmi důležité v návaznosti na paměťové a výpočetní úlohy aplikace. Rozlišení monitorů je na druhou stranu zásadní při vývoji aplikace, kde se musí pamatovat, pro jaké minimální rozlišení se aplikace navrhuje, ale zároveň by se nemělo zapomenout, že i monitory se můžou zničit, a v takovém případě je dost pravděpodobné, že by byly na určitý čas nahrazeny staršími monitory s menšími rozlišeními, a proto aplikace by měla být použitelná i na nich.

Tato situace znemožňuje využití aplikace Microsoft Access, která by byla vhodná pro daný problém. Proto zůstávají možnosti vytvoření buďto desktopové, nebo webové aplikace. Z důvodu náročnosti zpřístupnění a tvorby desktopové aplikace pro více uživatelů v síti je tato práce dále zaměřená na návrh a realizaci webové aplikace.

4. Návrh a realizace intranetové aplikace

4.1. Výběr technologií

Z hardwarových a softwarových možností MěÚ v Českém Těšíně a požadavků zaměstnanců je vyvozeno, že nejvhodnější varianta je webová aplikace. Jelikož je dostupný webový server, není nutné, aby aplikace byla nahrána na Internetu, ale postačí k ní mít přístup pouze lokálně na intranetu. Touto volbou, je vyřešeno pár problémů, které by mohly vzniknout, pokud by aplikace byla dostupná z internetu.

Prvním z nich je skutečnost, že funkčnost aplikace by byla plně závislá na funkčnosti internetu. To znamená, že pokud by došlo k útoku na MěÚ a bylo by nutné ho od internetu z bezpečnostních důvodů odpojit, pak by nebylo možné tuto aplikaci využívat.

Druhým problémem, který by mohl nastat, je bezpečnost samotné aplikace na internetu vůči intranetu. Na Internetu je mnohem vyšší hrozba a pravděpodobnost útoku ze strany miliard uživatelů a hlavně hackerů, kteří by mohli danou aplikaci napadnout. Oproti tomu na straně intranetu, při použití vlastního webového serveru, je hrozba mnohem menší, protože celá vnitřní síť je chráněna jak síťovým firewallem, tak firewally na jednotlivých stanicích.

Jako další důvod může být velmi obtížný přístup k aplikaci na vnitřní síti přes Internet. Je pravdou, že v mnoha organizacích a firmách je vhodné využívat interní aplikace i z jiných míst než z prostředí dané firmy, ale v případě webové aplikace na majetkovém odboru MěÚ tato podmínka neplatí.

Výběr tvůrce webové aplikace, pro řešení problému nebo inovaci, se také liší podle dostupných zdrojů. Pokud má zákazník méně finančních zdrojů, může využít nákupu už vytvořené aplikace od vývojářské firmy. Problém se může vyskytnout tehdy, pokud by byly požadavky příliš specifické a obecná aplikace by nesplňovala svůj účel. Pokud má spotřebitel dostatek finančních zdrojů a s aplikací nespěchá, je možnost využít vývoj aplikace na míru jeho požadavků[4].

V případě MěÚ je vývoj webové aplikace na míru více než vhodný a díky tomu, že daná aplikace je zásadní objekt v této práci a mezi požadavky je žádost o bezplatné vytvoření

[4] Více informací o možnostech inovace informačních systémů viz TVRDÍKOVÁ, Milena. *Aplikace moderních informačních technologií v řízení firmy: nástroje ke zvyšování kvality informačních systémů*. Praha: Grada, 2008. ISBN 978-80-247-2728-8, s. 36-37.

aplikace, pak její náklady na vývoj jsou nulové. Důležité je ještě zmínit, že při této variantě je zvolená její část centralizovaného zpracování. Tj. zpracování, které probíhá na straně serveru, a k němu jsou připojeny klientské koncové stanice.

Dál je tedy potřebné rozhodnout, jaký programovací jazyk se zvolí pro dosažení výsledku. Volba jazyka vždy záleží na vývojáři a taky na tom, jaké funkce má aplikace splňovat. V případě řešení tohoto problému je zvolen jazyk PHP díky tomu, že je nezávislý na platformě a vytvoření řešení, je při jeho použití, reálné. U tohoto projektu je navíc zvolen jeden z mnoha PHP frameworků, konkrétně Zend Framework.

Jak už bylo zmíněno dříve, umožňuje využívat mnoho připravených knihoven a je založen na MVC architektuře. Mezi jeho nevýhody patří to, že obsahuje mnoho kódů, které nejsou vůbec využívány, a proto zpracování jednoho požadavku může být oproti čistému PHP o dost pomalejší. Z tohoto důvodu se při použití tohoto frameworku využívá i tzv. opcode cache, kterých existuje více druhů. Pro tento konkrétní případ je zvolena možnost APC, která umožňuje zavolaný skript zkompilovat a uložit do vyrovnávací paměti (cache). Při příštím zavolání se už skript znova nekompiluje ale spouští se z paměti, což značně urychluje zpracování požadavku.

Volba mezi práci s frameworkem a čistým PHP je ovlivněna i tím, že práce v ZF je mnohem rychlejší než při programování komponent, které už naprogramované jsou a stačí je jen správně použít. Je pravda, že při programování bez knihoven se často využívá opravdu jen to, co se potřebuje, ale na druhou stranu musí programátor, pro vytvoření stejného výsledku, vynaložit mnohem více úsilí než v případě využití frameworku. Je pak logické, že firmy raději investují menší finanční náklady do hardware, pro zvýšení rychlosti aplikací, než do práce programátora, který by se snažil kód čím jak nejlépe optimalizovat. Tato volba firem častokrát znamená, že kód nebude nejčistější, ale budou minimalizované náklady.

Další možností, jak vyřešit tento daný problém, by bylo použití už dříve vytvořené OpenSource aplikace. U tohoto řešení je výhoda, že se nemusí dělat aplikace od začátku, ale použije se už dříve vytvořená a pouze se dolaďují detaily. Ale i tato možnost, oproti zvolené, má své vážné nevýhody. Jedna z největších nevýhod tohoto řešení je, že požadavky na aplikaci pro majetkoprávní odbor jsou velmi specifické a využité procento už dříve vytvořené aplikace by bylo mizivé. Využitelný by byl možná jen základ daného open source systému, ale zbytek specifických funkcí by stejně bylo nutné doprogramovat. Při tomto řešení by bylo nezbytné pochopit jádro systému a až následně by bylo možné se tuto aplikaci pokusit

vytvořit. Lze odhadovat, že přizpůsobení daného open source konkrétním požadavkům, by bylo velmi obtížné.

Následujícím rozhodnutím je volba databáze. Pro oddělení informatiky na MěÚ není typ databázového systému až tolik důležitý, ale je důležité, aby byl volně k dispozici a bez nákladů na provoz. A přesně takový je databázový systém MySQL, který je zvolen jako výchozí systém pro správu databáze dané aplikace. Další systémy nebyly vhodné, ať už z důvodu toho, že byly zpoplatněné, kapacitně omezené nebo funkčně značně zredukované.

4.2. Návrh aplikace

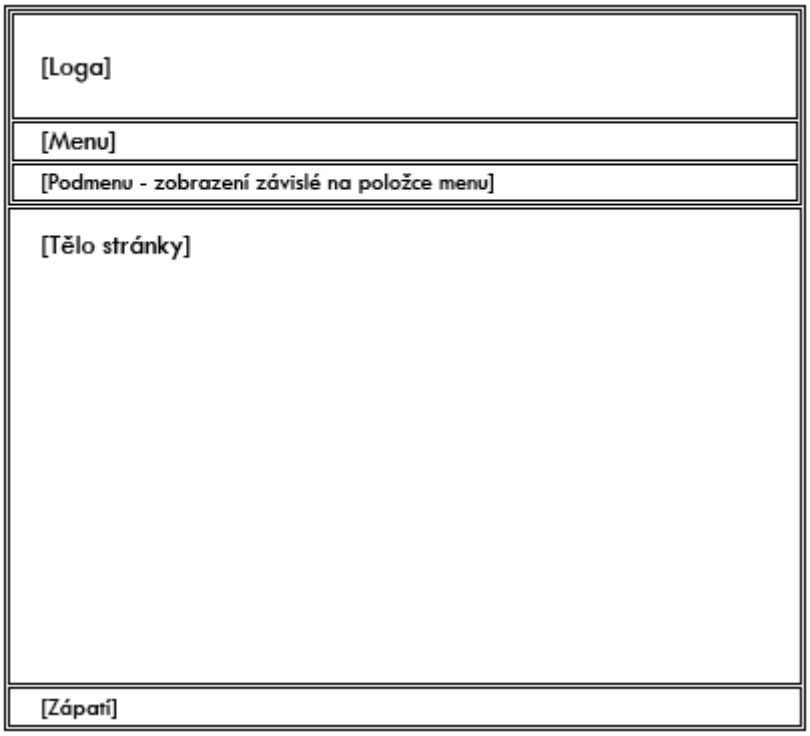
Při návrhu aplikace, je vhodné, aby se celá aplikace rozdělila na různé vrstvy a části, jelikož je její struktura více pochopitelná a lépe se na této struktuře aplikace vyvíjí, než když se vytváří ze spleti nápadů, kdy některé můžou být i dobré, ale může se na ně zapomenout nebo aplikace pak nemusí plnit původní záměr.

Z vlastních zkušeností vím, že grafická stránka aplikace je programátory velmi často opomíjena. Tato část návrhu je ale stejně důležitá, jako návrh datových modelů nebo databáze. Vhodně zvolené barvy, tvary a umístění prvků v layoutu stránky může první dojem z výsledné aplikace velmi zlepšit a také každodenní práci. Stejně tak špatně zvolený design může i sebelepší programátorskou práci s postupem času naprosto znehodnotit. Proto by vzhled aplikace a její funkčnost, či konzistence měly jít ruku v ruce.

Rozložení stránek je zvoleno podle zažitých standardů. Dodržování těchto konvencí může být prospěšné pro orientaci na webu. Naopak vytvoření svého vlastního rozložení může být matoucí a pro uživatele nepřehledné. Mezi nepřehledné aplikace by se mohly řadit ty, které mají menu stránky uprostřed či dole a při rozsáhlejších stránkách by pro zobrazení menu bylo dokonce nutné stránku srolovat úplně dolů. Zvolené řešení je proto složeno z hlavičky, která obsahuje logo aplikace a logo města a je umístěna v horní části stránky, hlavního menu a podmenu, které slouží k orientaci na stránce. Následuje tělo stránky a pak zápatí. V zápatí jsou informace o autorských právech a licencích na různé technologie využité právě v aplikaci.

Šířka rozložení se přizpůsobuje rozlišení monitoru, tzn. je nastavená tak, aby při zmenšování okna prohlížeče se zmenšoval i obsah těla stránky. Pokud obsah těla stránky už nemůže být menší a přesto se hodnota šířky stránky zmenšuje, zobrazí se u těla stránky posuvník. Pokud je šířka stránky menší než minimální šířka stránky, pak se i za zápatím

stránky zobrazí posuvník. Takovýmto řešením se eliminují potíže jako různé rozhození elementů stránky, nesouladnost částí apod.



Obr. 4-1 Rozložení stránek

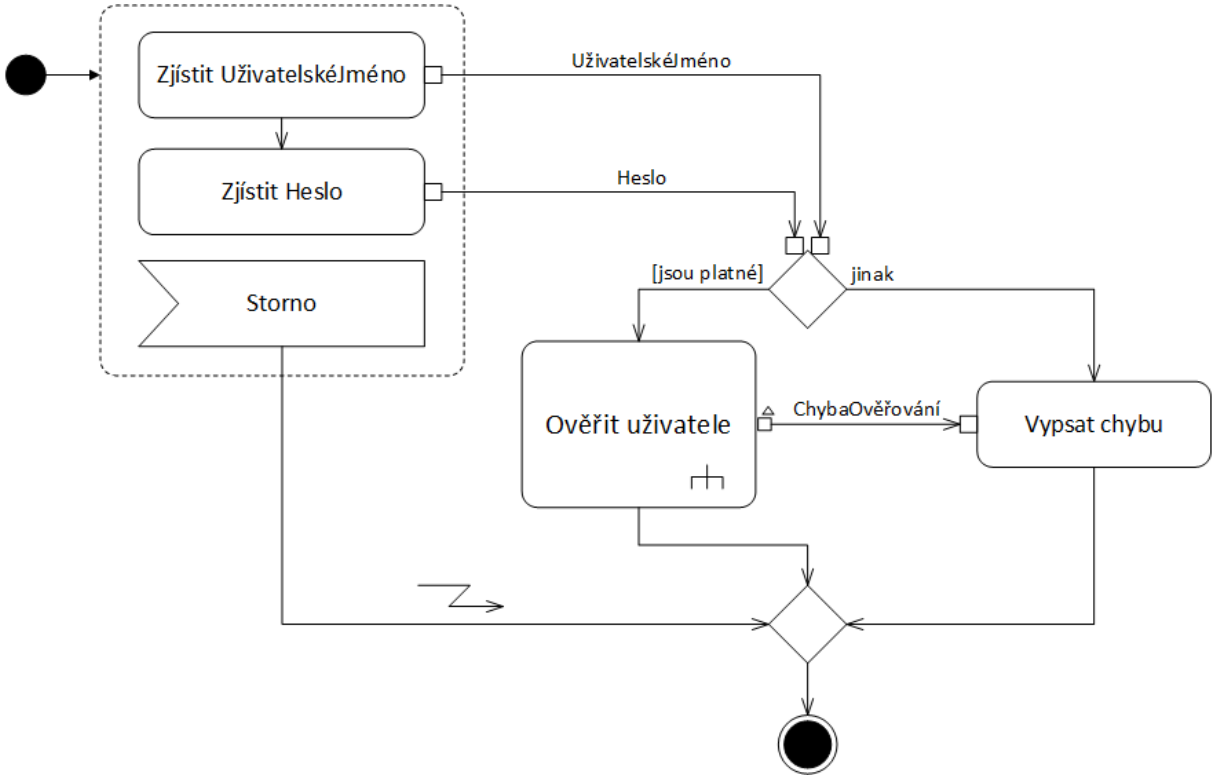
Pro danou aplikaci jsou zvolené barvy, které lze vidět běžně v kanceláři nebo na kancelářských papírech. Mezi tyto barvy patří černá a bílá, jakožto velmi dobře kontrastní barvy, dále světle a tmavě oranžově-červená, pro zvýraznění odkazů, kterou je možno vidět jak na kancelářských šanonech, tak i při zvýrazněném textu. Další barvou je šedá a její odstíny pro klidný vzhled.

Obr. 4-2 Základní barvy aplikace

V další fázi návrhu je důležité přejít k návrhovým modelům. Tyto modely slouží k přiblížení procesů, funkcí a dat uvnitř aplikace. Nad těmito modely se následně realizuje aplikace, a proto by se neměly opomíjet, ani zanedbávat. Při tvorbě návrhových diagramů se často poukáže na možné sémantické chyby, které by při dokončování realizace mohly být velmi obtížné na opravu. Modely často můžou sloužit i pro zorientování se ve struktuře aplikace dalším vývojářům, kteří se snaží buďto aplikaci inovovat nebo některé její části opravit.

4.2.1. Proces přihlašování

Důležitým procesem většiny moderních aplikací je přihlašování uživatelů. Stejně je to i v případě řešeného problému. Na následujícím modelu můžeme vidět chování systému, když se uživatel pokusí přihlásit do aplikace. Pro úspěšné přihlášení je nutné zadat validní údaje, a ověřit, zda se uživatel s příslušným heslem nachází v databázi. Jestli jsou všechny podmínky splněny, je uživatel v systému přihlášen. Pokud ne, je uživateli zobrazena zpráva s chybou a proces přihlašování je počítán za neúspěšný.



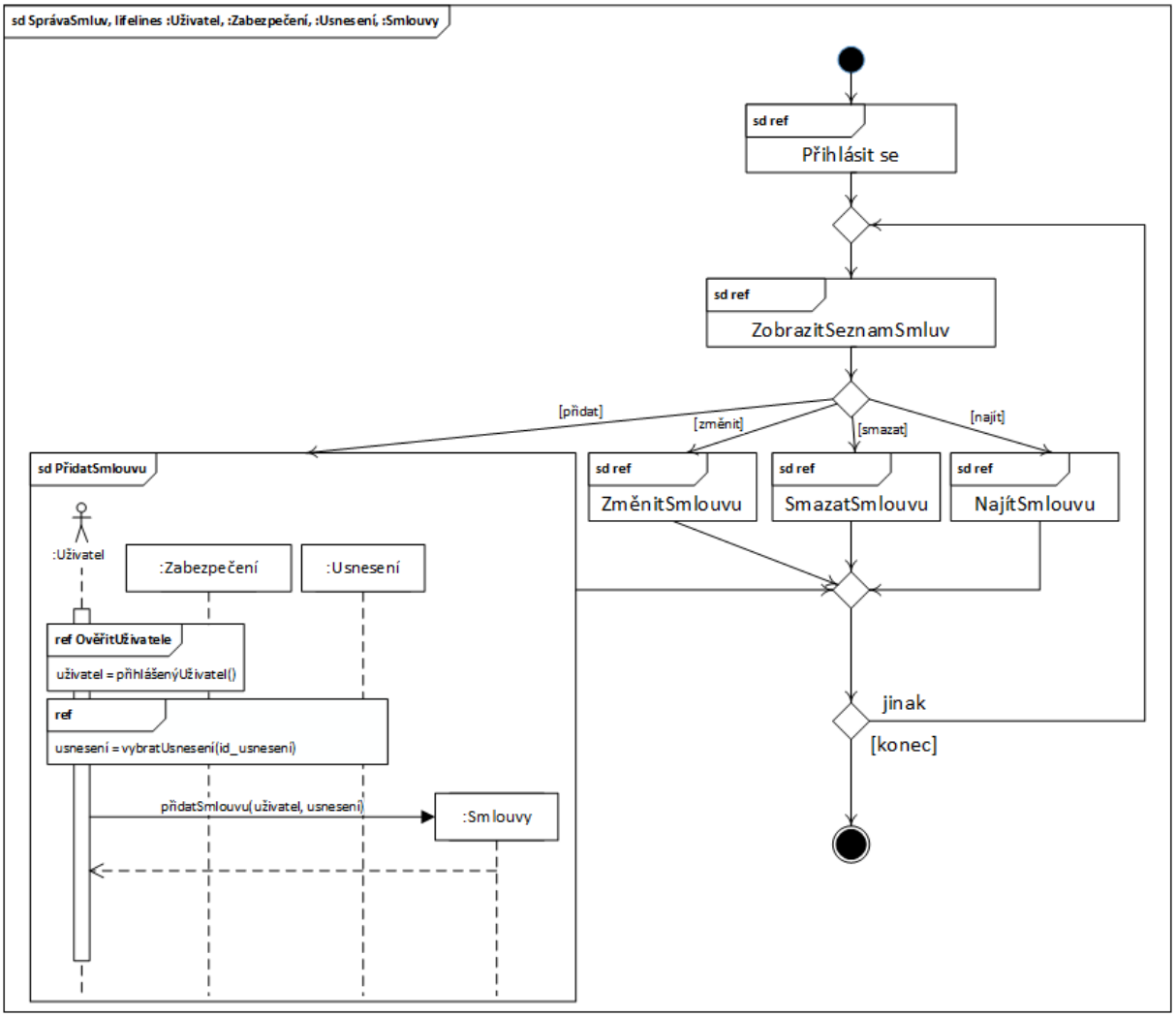
Obr. 4-3 Diagram přihlašování

Tento model i model popsaný níže, jsou modely v jazyce UML (Unified Modeling Language) a patří do skupiny diagramů aktivit. Tyto diagramy se vyznačují tím, že umožňují modelovat procesy jako aktivity, které jsou složeny z množiny uzlů spojených jednotlivými hranami. Diagramy zobrazují chování operací při různých podmínkách. Jsou velmi často požívanými ve fázi návrhu aplikace, ale lze je použít i ve fázi analýzy.

Díky značné flexibilitě je možné podle metodiky Unified Process využívat diagramy aktivit pro různorodé procesy. Ve fázi návrhu je jejich nejčastější využití při detailním modelování procesu a při podrobných algoritmech. Diagramy aktivit jsou většinou velmi snadno srozumitelné z důvodu, že zúčastnění se v minulosti mohli mnohokrát setkat s různými vývojovými diagramy a proto jsou dobrými komunikačními prostředky.[21]

4.2.2. Diagram správy smluv

Dalším procesem, který je zachycen v diagramu aktivit, je správa smlouvy s detailním zaměřením na přidání nového záznamu smlouvy. Smluv je v této situaci sedm typů, ale přestože mají odlišené vnitřní logické struktury, je možné pomocí diagramu aktivit vhodně navrhnout proces správy smlouvy.



Obr. 4-4 Diagram správy smluv

Proces změny, smazání nebo hledání daného typu smlouvy je skoro totožný jako přidání smlouvy nové, jenom s rozdílem, že se údaje v záznamu smlouvy aktualizují na základě identifikátoru smlouvy a i soubor smlouvy s cestou k jeho zobrazení je zaměněn. Jelikož to nemá na model žádný vliv, byla jako vhodnější a přehlednější zvolena varianta zobrazení procesu správy smlouvy v diagramu aktivit s vnitřním sekvenčním diagramem.

Na začátku procesu musí být uživatel přihlášen. Po zobrazení daného typu smluv je mu umožněno smlouvy přidávat, měnit, mazat a hledat. Přidávání, hledání záznamů smluv

a zobrazování souborů smluv je umožněno všem uživatelům. Změna a odstranění smlouvy je uživateli dovoleno právě tehdy, jestliže má roli správce aplikace nebo danou smlouvu už dříve vytvořil. Při přidávání smlouvy je pro uživatele nutností být přihlášen, a pokud je, uloží se jeho identifikátor do proměnné. Uživatel následně vybere usnesení ze seznamu usnesení, které se týká nově přidávané uzavřené smlouvy a identifikátor usnesení se uloží do další proměnné. Tyto dvě proměnné jsou následně využity při přidávání smlouvy spolu s dalšími vstupními hodnotami a tímto je proces přidání smlouvy u konce.

Uživateli je poté opět umožněno provádět operace nad aktualizovaným seznamem smluv jako přidání, změna, smazání a hledání konkrétní smlouvy. Pokud se uživatel rozhodne již více operací se smlouvami v dané době neprovádět, je tím ukončen proces správy smluv.

4.2.3. Struktura databáze

Při návrhu databázové struktury, je důležité vytvářet objekty databáze s pohledem na normální formy, ale zároveň je důležité se soustředit na požadavky zaměstnanců. Konkrétně brát v úvahu převod evidenčních sešitů do elektronické podoby.

Problémy vznikají, pokud je snaha o převedení objektového vztahu do relačního, konkrétně u generalizace, tj. nadtypů a podtypů. Jedná se např. o objekty fyzické a právnické osoby, které mají společné vlastnosti jako např. číslo popisné, ulici a město. Pak fyzické osoby se vyznačují jménem, příjmením a datem narození, kdežto právnické osoby, v lepším případě, jsou charakterizovány číslem IČO a názvem organizace. Řešení této situace se nabízí tři. Za prvé lze vytvořit jeden objekt osoba a zahrnout do něho jak společné vlastnosti, tak vlastnosti charakteristické pro jednotlivý typ osoby. Za druhé, lze vytvořit objekty dva, jeden pro fyzickou osobu a druhý pro právnickou, kde jak fyzická, tak právnická osoba bude obsahovat společné hodnoty číslo popisné, ulice a město. Za třetí, je možno vytvořit tři objekty, které budou reprezentovat tři reálné objekty. V relačních databázích je potřeba však tyto vztahy ošetřit triggery, které se budou starat o zachování integrity.

Výhodou prvního řešení je značná rychlost při přístupu k datům, ale nevýhodou je mnoho prázdných hodnot. Mezi přínosy druhé varianty patří opět rychlost a tentokrát i minimalizované prázdné hodnoty, ale záporem je značná redundance dat. Ve třetím řešení je výhoda v tom, že je to plně objektový přístup, ale nevýhody jsou pomalá rychlost a nutnost ošetření vztahů triggery.

V případě této práce se vyskytují tyto problémy u tří objektů. Mezi tyto objekty patří smlouvy, nemovitosti a osoby. Každá ze tří variant možností řešení je vhodná pro jednotlivé

případy. První varianta, sloučení objektů do jednoho objektu, je vhodná pro objekty smlouvy a nemovitosti, z toho důvodu, že jak smlouvy, tak nemovitosti obsahují mnoho podtypů. Čím více podtypů daný nadtyp obsahuje, tím pomalejší by byl přístup při variantě ošetřené triggery a jelikož redundance dat je velmi nevhodný stav, je zvolena první varianta. Redundance dat může ohrozit konzistenci dat v případě, že bude nutné data změnit. Při změně je nutné pamatovat na změnu všech stejných dat, a proto se velmi často může stát, že se data zaměnit zapomenou. Řešením tohoto typu problému by bylo opět využití triggerů, které by se spouštěly při změně nebo smazání. Dalším objektem jsou osoby. U tohoto objektu je vhodnější zvolit variantu, která nejpřesněji zachovává objektový model, a vztahy jsou ošetřeny triggery. Tato varianta je ze všech nejvhodnější, jelikož podtypy osoby jsou pouze dva a to právnická a fyzická osoba. Proto zpomalení při čtení není tak markantní, jako by to bylo v případě daných dvou předchozích objektů.

Z finančních důvodů je pro danou aplikaci zvolený databázový systém MySQL s úložištěm dat InnoDB. Toto úložiště podporuje mimo jiné práci s cizími klíči a triggery, a proto je pro dané řešení nejvhodnější.

Na základě analýzy požadavků a návrhu databáze je vytvořen datový model relací mezi jednotlivými entitami. Aplikace je postavena na jedenácti relačních tabulkách, u které jsou identifikovány vždy primárním klíčem. I v případě možných slabých entit, je jako identifikátor zvoleno automaticky generované přírůstkové číslo. E-R diagram je vyexportován z aplikace MySQL Workbench s UML notací vztahů (viz. Příloha č. 1: E-R diagram). Další důležitou částí při realizaci databáze jsou specifikace domén.

V této fázi se určují názvy atributů, jejich typy, délka, přípustnost null hodnot a popis. Důležitou části specifikací domén je určení primárních a cizích klíčů. Specifikace domén se zapisuje formou tabulek a její účel je, aby datový model byl pružný pro implementaci budoucích změn, které by byly způsobeny možným budoucím vývojem. V této práci jsou vytvořené specifikace domén ze všech jedenácti tabulek (viz. Příloha č. 2: Specifikace domén).

4.2.4. Triggery

Jsou to procedury, které jsou spouštěny při vkládání, změně nebo mazání záznamů z tabulky, nad kterou jsou vytvořeny. Jak už bylo zmíněno dříve, jejich významem je zachování konzistence a zamezení redundance dat.

V případě této práce jsou za potřebí triggery dva - jeden pro fyzické a druhý pro právnické osoby. Fyzická a právnická osoba, jsou v tomto konkrétním případě subtypy entity osoba. Při vložení záznamů dané fyzické osoby se musí prvně vložit záznam do entity „osoba“ a následně se stejným identifikátorem jako cizím klíčem, do entity „fyzicka“. Problém by mohl vzniknout, pokud by uživatel vkládal osobu právnickou a nedopatřením by využil nebo chybou aplikace, identifikátor pro právnickou osobu z nadtypu osoby, který by už byl primárním klíčem pro osobu fyzickou.

Naskytuje se vícero řešení. Řešení jak přes prezentační nebo logickou vrstvu. Jelikož by ale tento problém mohl značně narušit strukturu databáze, je vhodné ošetřit hlavně datovou vrstvu, a pokud možno i vrstvy ostatní.

Z tohoto důvodu je pro zabezpečení datové vrstvy vytvořený trigger, který se spustí před každým vložením nového záznamu do obou tabulek (viz. Příloha č. 3: Trigger fyzických osob a výsledek nesprávného vložení). Jak už název přílohy napovídá, v příloze není k nalezení pouze struktura vytvořeného triggeru pro fyzické osoby, ale i výsledné hlášení při pokusu o vložení nesprávného identifikátoru. V příloze je možno vidět pouze trigger s chybným hlášením pro fyzické osoby, ale u právnických osob se řešení příliš neliší od řešení u osob fyzických.

4.3. Realizace programu

Při tvorbě je důležité si určit jednotlivé kroky, které tvoří osnovu realizace. Tyto kroky se následně naplňují, až je v konečném důsledku aplikace hotová. Není vhodné začít další krok bez dokončení předchozího, jelikož se můžou začít objevovat zbytečné chyby a projeví se nejčastěji na časové prodlevě, která může znamenat různé penalizace ze strany zadavatele apod. Jako jednoduché kroky při vývoji aplikace s MVC strukturou můžou být vytvoření modelu, následné vytvoření řadiče a nakonec pohled. Další důležitou stránkou je časté zálohování a nejlépe na více paměťových médií.

Vývoj aplikace v Zend frameworku umožňuje využívat různé možnosti a komponenty. Jedním z mnoha modulů je CLI Tool, který umožňuje snadné a rychlé vytváření MVC části díky příkazovému řádku. Pro vývojáře je časově náročné psát stejný kód nespočetně krát, a proto je tento nástroj velkým přínosem (viz. Příloha č. 4: CLI Tool ve Windows PowerShell).

Díky tomuto nástroji se automaticky generuje základní kód řadičů, modelů, pohledů, formulářů a mnoho dalších komponent. Je samozřejmostí, že kód je generován podle frameworkových konvencí a proto si na ně vývojář může lépe navyknout.

4.3.1. Modely

V modelové části MVC architektury se zprostředkovává přístup k datům nejčastěji uloženým v databázi, ale také se definují formuláře a s nimi spojené vlastnosti. Pro aplikaci je nutno vytvořit modely pro všechny fyzické tabulky, tedy pro ty, se kterými má aplikace pracovat. Databázové modely v ZF rozšiřují třídu `Zend_Db_Table_Abstract` a jsou fyzicky uloženy ve složce „DbTable“, která se nachází v adresáři „models“. Jelikož k modelům patří i správa záznamů tabulek, jako např. čtení, zápis, změna nebo odstranění záznamů, je dobré, aby se funkce ve všech databázových modelech neopakovaly. Díky tomu, lze vytvořit modelovou třídu, která nerozšiřuje žádnou jinou třídu a slouží, jako výchozí třída při práci s databázovými modely. Této modelové třídě se do konstruktoru nastaví parametr instance `Zend_Db_Table_Abstract` a následně lze při vytvoření její instance využívat všechny funkce pro všechny databázové modely. Mezi tyto funkce se neřadí pouze čtení, zápis, změna a smazání dat tabulky, ale taky jakékoli podpůrné funkce.

Jak už bylo zmíněno dříve, k modelům patří i formuláře. Formuláře mají úzký vztah k databázovým modelům, jelikož právě data z formulářů můžou být následně vkládány přes databázové modely do tabulky. Proto je velmi časté, že formulářů je stejně nebo víc než databázových modelů. Formuláře v ZF rozšiřují třídu `Zend_Form`, která umožňuje využívat už před vytvořenou strukturu formu a následným přidáním jednotlivých prvků, je formulář dokončený. Ke každému prvku formuláře lze využít už vytvořených validátorů a filtrů, ale taky vlastních validátorů nebo filtrů.

V případě této aplikace se mnoho vlastností prvků opakuje. Proto je vhodnější vytvořit svůj vlastní formulářový prvek, který bude rozšiřovat výchozí prvek, ale bude nést své konkrétní vlastnosti. Takto vytvořené formulářové prvky se ukládají do knihovny aplikace s vytvořeným adresářem.

Díky atributů tříd jednotlivých prvků formuláře je možno použít pluginy, např. pro vytvoření masky při zadávání hodnot uživatelem, nebo možnost naformátovat text v rámci html prvku „textarea“ apod. Právě tyto dva pluginy jsou použity v aplikaci týkající se této práce. Pro vytvoření masky vstupu je využíván plugin „MaskedInput“ a pro zobrazení možností formátování prvku textarea je použit plugin „CKEditor“. Oba pluginy jsou sice

závislé na technologii JavaScript, ale při vypnutí je i tak kontrolována hodnota vstupu podle regulérních výrazů a bez formátování text v „textarea“ se přinejhorším lze taky obejít. Plugin masky pro omezení zadávání znaků se využívá ve formulářích při přidávání nových identifikátorů spisů, usnesení, všech druhů smluv, parcely u nemovitosti a nebo IČO u právnické osoby. Formátování textu se využívá ve formulářích nahrání usnesení nebo smlouvy jako textu, který lze do daného okna zkopírovat i s formátováním. Tyto formuláře slouží k převedení naformátovaného textu do souboru formátu PDF.

Na každé stránce, kde se vyskytuje tabulka, je vytvořený i formulář pro řazení dat. Tento formulář se automaticky plní daty (sloupci) podle dané tabulky. Formulář neumožňuje pouze řazení dat, ale taky vyhledávání podle celých řetězců i jejich příslušných částí.

V rámci validací formulářů se využívají validační třídy „NotEmpty“ v případě povinného vyplnění, „RegExp“ pro řetězec shodný s regulérním výrazem, „StringLenght“ pro určení minimální a maximální délky zadaného řetězce, „StripTags“ k odstranění většiny html značek, „StringTrim“ pro odstranění počátečních a koncových mezer, „Unique“ ke zjištění, zda je daná hodnota v příslušném sloupci unikátní apod.

4.3.2. Řadiče

Tato komponenta se stará o změny v modelech nebo pohledech při požadavku uživatele. Dokáže číst vstupy z HTTP požadavků, používat funkce z modelů, a dokáže komunikovat skrze požadavky s pohledem. Každý řadič v aplikaci se skládá z inicializační metody `init()`, výchozí akce „index“ a jednotlivých akcí, které k danému řadiči patří. Tyto akce následně zobrazují stejně pojmenované pohledy.

V aplikaci je vytvořeno dvacet dva řadičů. Řadič „Index“ zobrazuje hlavní stránku s informacemi o aplikaci. Řadič „Uzivatele“, slouží ke správě uživatelů. K tomuto řadiči má přístup pouze správce aplikace. Další řadit nese název „Ukoly“. Tento spravuje úkoly všech uživatelů a umožňuje správci aplikace vkládat úkoly pro běžné uživatele. „Spisy“ spravují záznamy o spisech majetkoprávního úřadu. Uživatele můžou měnit jen ty záznamy, které sami vytvořili s výjimkou správce a záznamů fyzických a právnických osob. Řadič „Usneseni“ má na starosti správu o usneseních schválených Městem Český Těšín. Jedna smlouva může mít vztah k více usnesením. Dalším nese název „Smlouvy“, který spojuje dalších sedm řadičů podle jednotlivých typů smluv. V neposlední řadě je řadič „Nemovitosti“, který spravuje záznamy o nemovitostech, které mají určitou souvislost s uzavíranými smlouvami. Taktéž řadiče pro fyzické i právnické osoby spravují záznam o osobách ve vztahu

k uzavíraným smlouvám. Pak je vytvořen řadič pro autorizaci uživatelů, ale taky pro odchytávané chyby aplikace.

Mezi nejčastěji používané akce jednotlivých řadičů, při práci nad databázovými tabulkami, se považují akce „pridat“, „zmenit“ a „smazat“. Jedná se o akce spojené s přidáváním, změnou a mazáním záznamů z tabulek.

Stejně tomu je i v případě této vypracovávané aplikace. Obecně v aplikaci je většina řadičů velmi podobných, proto je vhodné jednotlivé akce více popsat. Na začátku je potřeba nastavit vhodné data v inicializační proměnné. Mezi tyto data se počítá instance modelu pro daný řadič, pole názvů sloupců, které budou použity ve formuláři pro řazení a vyhledávání a identifikátor přihlášeného uživatele.

První je akce s označením „index“. Tato akce vytvoří formulář pro vyhledávání a řazení, a následně ho zpřístupní pohledu. Z proměnné typu aplikace je získána paměť cache typu „Core“, a pokud je v proměnné požadavku nastavený identifikátor, tak je cache vypnuta, jinak je zapnutá. Z aplikační proměnné získáme druhý typ paměti cache, tentokrát typ „Output“ a zavoláme „action helper“[5], který zpracovává formulář řazení a hledání v tabulce. Pokud byl odeslán formulář pro řazení nebo vyhledávání a nastala chyba, je zobrazena. Další je podmínka, která ověřuje, jestli uživatel je správce aplikace a pokud ne, přidá k výslednému výběru i sloupce s uživateli, kteří mají k těm daným záznamům vytvořený vztah. Následně vypne první typ cache a pošle pohledu zprávu, že uživatel není správce a může upravovat pouze ty záznamy, které sám vytvořil. Potom je využita metoda „zobraz“ modelového vzoru a v ní se odešlou parametry s typem cache a dalšími údaji o výsledném výběru. Na konci akce „index“ se zase povolí cache typu „Core“.

Druhá akce nese název „pridat“. U této akce se vytvoří instance formuláře pro přidávání záznamů a načte se příslušná session se záznamy, které mohly být použity už dříve. Pokud je nastaven parametr s názvem identifikátoru, a příslušná session není prázdná, naplní se formulář daty z proměnné rozsahu session. Následně se ověřuje, zdali je požadavek typu POST. Pak je podmínka, že pokud bylo stisknuto tlačítko vybrat, tak se všechny hodnoty formuláře uloží do dané session a stránka se přesměruje na jiný řadič, kde proběhne vybrání daného záznamu. Pokud je formulář validní, uloží se hodnoty formuláře pro proměnné „hodnoty“ a tyto hodnoty se odešlou metodě „pridat“ modelového vzoru pro přidání záznamů

[5] Třída, která umožňuje redukovat kód aplikace tím, že obsahuje tento znovu použitelný kód a je možno ji volat z jakéhokoli řadiče aplikace.

do tabulky. Pokud není validní, řadič přepošle chybové zprávy do pohledu a formulář zůstane vyplněný. V případě, že se záznamu týká i konkrétní fyzický soubor, tak se ověřuje, jaký typ nahrání souboru uživatel zvolil a podle toho je přesměrován na patřičnou stránku. Typy nahrání souborů jsou celkem dva. Prvním typem je nahrání textu, který je následně systémem převeden do formy PDF dokumentu a druhým typem je nahrání přímo konkrétního souboru jako přílohy.

Jako třetí akce se v aplikaci nejčastěji využívá akce „zmenit“. Tato akce je skoro totožná s akcí „pridat“, jenom s pár rozdíly. Jedním z rozdílů je, že se při spuštění akce ověřuje, zda li je nastavený parametr s korektním identifikátorem, a pokud ano, je formulář naplněn daty odpovídajícími danému identifikátoru. Další důležitou změnou je, že se neprovádí metoda „pridat“ modelového vzoru, ale „zmenit“ která změní záznam v databázové tabulce podle identifikátoru.

Další a častokrát poslední akcí je „smazat“. Při spuštění této akce je podmínka, jestli je požadavek typu POST. Pokud ne, použije se identifikátor z parametru GET, který pohledu zpřístupní data týkající se tohoto daného identifikátoru. Pokud je požadavek typu POST, a bylo stisknuto potvrzovací tlačítko pro smazání „Ano“, tak se následně zavolá metoda „smazat“ modelového vzoru s vnitřním parametrem identifikátoru. Po provedení ať už přidání, změny nebo smazání je stránka vždy přesměruje na výchozí akci, tj. „index“. Výjimkou mezi akcemi může být počítána akce s názvem „vybrat“, kdy je požádán model o tabulková data a tyto data slouží uživateli k přehlednějšímu výběru relačních záznamů. Po výběru daného záznamu se stránka přesměruje na příslušnou akci.

Výjimkou mezi řadiči je řadit s označením „Auth“. Tento řadič ověřuje zadané vstupní jméno a heslo se záznamy v databázi. Pokud se jméno a heslo shoduje, uloží objekt uživatele do proměnné rozsahu session a přesměruje na řadič „Index“. Pokud se údaje neshodují, je poslána chyba pro zobrazení v pohledu a stránka zůstává stejná.

4.3.3. Pohledy

Pohledy slouží k výslednému zobrazení informací a dat na obrazovky uživatelů. Tyto informace zpřístupňuje vždy řadič, konkrétně jeho akce pro příslušný pohled. V ZF jsou pohledy fyzické soubory s příponou „phtml“. Při generování jednotlivých pohledů se vždy jako první generuje layout stránky a následně až obsah pohledu. Při generování layout stránky se generuje postupně hlavička s externími styly a metadaty, tělo a patička stránky s příslušnými náležitostmi.

Nejčastěji používané pohledy v aplikaci nesou názvy totožné s nejčastěji používanými akcemi. U pohledu „index“ v případě, že stránka využívá jazyka JavaScript, nahrají se externí soubory knihovny jQuery, ale taky vlastní externí JS soubory. Potom se zobrazí tlačítko „Přidat“ a opět pokud je povolený v prohlížeči JS, tak se zobrazí tlačítko pro zobrazení všech záznamů a nastavení. Pokud povolený není, zobrazí se formulář nastavení (tj. formulář pro vyhledávání a řazení záznamů) automaticky a tlačítko pro zobrazování všech záznamů a zobrazování nastavení se skryje. Když je pak nastavena zpráva pro pohled, tak se tato zpráva vypíše. Pokud nastavená není, nezobrazí se. Nakonec následuje tabulka s hodnotami přijatými od řadiče.

V případě pohledu „pridat“ se v obsahu změní nadpis stránky, načtou se příslušné externí skripty a vykreslí se formulář pro přidání nového záznamu. Pokud byl formulář vyplněn chybně, u jednotlivých položek se zobrazí chybové hlášení. Podobný scénář se opakuje i při změně záznamu. S rozdílem, že formulář se nevykreslí prázdný, ale naplněný konkrétními hodnotami odpovídajícími zvolenému identifikátoru. V pohledu „smazat“ se opět zobrazí nadpis a pod nadpisem se objeví text, přibližující uživateli informace o záznamu, který může být smazán. Pak jsou zobrazeny dva tlačítka „Ano“ a „Ne“ pro rozhodnutí uživatele, zda chce daný záznam smazat, či ne.

Zajímavostí v aplikaci může být formulář pro vyhledávání a řazení dat. V případě zapnuté podpory JS je jeho zobrazení závislé na akci tlačítka „Zobraz nastavení“. Výchozí stav formuláře by byl buď na všech stránkách skrytý, nebo na všech stránkách zobrazený. Zajímavostí je, že skrytý formulář zůstává skrytý a zobrazený formulář zůstává zobrazený při změně stránky. Tohoto cíle bylo dosaženo s využitím JS proměnné `window.name`, která je nastavitelná v JS kódu a její hodnota se při změně stránky nezmění.

Mezi výhody vývoje psaní aplikace v ZF patří i využívání tzv. „View Helperů“(dále jen helperů). Tyto helpery jsou třídy, umožňující využívat stejný kód mezi všemi pohledy. Tyto helpery v aplikaci obsahují kód pro formátování data, formátování ceny, zobrazení přihlášeného uživatele apod.

Při vývoji aplikace je využito front-end frameworku Bootstrap, pro rychlejší a snadnější grafický vývoj aplikace. Tento rámec má předem připravené, graficky zpracované, formulářové prvky, které lze velmi snadno využít při realizaci vzhledu výsledných stránek. Kromě formulářových prvků obsahuje i styly pro písma, tabulky, obrázky, ikony apod. Jeho další využití je v JS. Obsahuje předem vytvořené skripty, které se velmi jednoduše

na stránkách používají. V případě řešené aplikace jsou využity pouze formulářové prvky s ikonami a skript pro změnu vzhledu tlačítka nahrávání souboru. Výsledný design aplikace je proto mnohem modernější a přehlednější (viz. Příloha č. 5: Stránka „usnesení“ a příslušné akce).

4.3.4. Optimalizace

Rychlost aplikace je dalším důležitým aspektem při její realizaci. Optimalizace rychlosti je proces testování, upravování a dalšího testování rychlosti výsledné aplikace. Pro zjišťování rychlosti řešené aplikace je využito freewarových programů a pluginů jako FireBug, ZFDebug a XDebug. Všechny tyto komponenty měří rychlost jiným způsobem, a proto je lepší se při optimalizaci zaměřit na jeden z těchto nástrojů. Dalším poznatkem je, že všechny tři nástroje ukazují zkreslenou rychlost načítání stránky, jelikož při jejich zaktivování je zobrazování vždy o něco pomalejší oproti skutečnosti, jelikož i dané měření rychlosti způsobuje určité zpomalení. Dalším faktorem ovlivňující rychlost je i rychlost serveru (při zpracování požadavku) a rychlost samotného prohlížeče (při načítání skriptů nebo zobrazování jednotlivých prvků stránek). Proto u různých prohlížečů se ve skutečnosti může naměřit různá rychlost zpracování a zobrazení a v konečném důsledku je výsledná, skutečná rychlost velmi těžce měřitelná.

Zobrazení tabulky ve webové aplikaci bez žádné optimalizace se třemi záznamy trvá v průměru 1462,2ms (při 5 měřeních programem WinCacheGrind používající XDebug). Taková doba je naprosto nevyhovující, jelikož by velikost ušlé mzdy při čekání na zpracování požadavku byla větší než finanční přínos aplikace. Pro správnou optimalizaci je vhodné provádět kroky, které jsou popsány v příručce ZF pro optimalizaci na jejich stránkách[6]. Po splnění všech těchto kroků, je vhodné zaměnit v konfiguračním souboru application.ini položku host databáze z „localhost“ na „127.0.0.1“. Při použití „localhost“ se MySQL knihovna pokouší připojit na místní soket, místo použití protokolu TCP/IP, což značně zpomaluje rychlost zpracování. Mezi další rady patří využívání, už dříve zmíněné, op code cache nejlépe APC a ukládat do paměti cache vše, co do ní ukládat lze a příliš často se nemění. Při přílišných aktualizacích by se muselo provést pokaždé vymazání paměti a znovu nahrání, což by vedlo opět ke zpomalování aplikace. ZF umožňuje využívat mnoho typů cache, ale pro tuto aplikaci jsou zvolené pouze dva. Cache „Core“ je paměť cache, který je vhodný pro ukládání do paměti různých typů proměnných. Mezi tyto proměnné se může řadit

[6] Více informací na: http://framework.zend.com/manual/1.12/en/performance.html

i výsledné pole hodnot z tabulky nebo tabulková metadata. Druhým použitým typem je cache „Output“, která umožňuje ukládat do paměti jakýkoliv zachycený výstupní obsah, který se nachází mezi metodami `start()` a `end()`. Mezi takovýto obsah můžeme považovat jak celé neměnné části layoutu, tak i formuláře apod. Zkoušet optimalizaci s ukládáním „core“ i „output“ cache je vhodné až po dokončení aplikace, jelikož při jejím vývoji, můžou nastávat různé chyby, které ale díky cache můžeme přehlédnout.

Po optimalizaci stejného typu tabulky se stejným počtem záznamů je rychlost zpracování v průměru 241,2ms (při použití stejného nástroje na měření jako dříve). Tato rychlost, jak bylo zmíněno dříve, není skutečná rychlost zobrazování stránky, ale je ukazatel o kolik procent se zvýšila rychlost zpracování požadavku. V tomto konkrétním případě se rychlost zvýšila o přibližně 600%.

Optimalizace je proces vhodný nejen kvůli tomu, že vede ke zrychlení a optimalizování aplikace, ale taky proto, že při optimalizování se objeví mnoho skrytých chyb, na které by se při vývoji mohlo zapomenout. Proto je při optimalizaci ušetřen čas, který by byl vynaložen při testování aplikace.

4.3.5. Bezpečnost

Ve fázi realizace je nezbytné provést dostatečné zabezpečení aplikace. Za nejběžnější útoky na webové aplikace lze považovat útoky jako SQL injection, XSS (Cross-site scripting) a CSRF (Cross-site request forgery). SQL injection je útok, který napadá neošetřená vstupní pole, do kterých se útočník snaží vložit své vlastní SQL příkazy. XSS je útok založený na vsunutí kódu do záznamu aplikace. Nebezpečí vloženého skriptu pak může ohrožovat všechny potenciální uživatele. Útočník použije útok CSRF, když se snaží o pozměnění požadavků legitimního uživatele, např. pro administrační úkony, ke kterým by měl mít práva pouze uživatel ve skupině administrátorů, možné změny struktury, zjištění citlivých údajů apod.

Je důležité se v řešené aplikaci zaměřit na zabezpečení právě proti těmto typům útoků. Pro zabránění útoku typu SQL injection je vhodné používat, vždy když je používán uživatelský vstup, buďto využívat „placeholders“, kdy všechny vložené data jsou obaleny v jednoduchých uvozovkách, a ZF se postará o vyčištění pochybných vstupů nebo totéž platí při využití metody `quoteInto()`.

Při zabezpečování proti XSS útoku, je vhodné využít ZF filtr „StripTags“. Jenže tento filtr, stejně jako stejnojmennou PHP funkci, už dokáží útočníci obejít a proto je nezbytné při

jakémkoli vypisování dat z databáze využívat ZF metodu `espace()`, která volá PHP metodu `htmlspecialchars()` pro převedení nebezpečných znaků přímo ze záznamů v databázi do html entit v podobě řetězců.

Pro zabránění útočníkovi v útoku CSRF je vhodné využít už některý z existujících pluginů nebo vytvořit plugin vlastní, který přidává každému formuláři při jeho vygenerování jedinečný token a tento token uloží do proměnné rozsahu typu session. Při požadavku o zpracování daného formuláře je daný formulářový token porovnáván s tokenem uloženým v session a pokud se tyto tokeny neshodují, pak se jedná s největší pravděpodobností o útok CSRF. Není tomu vždy tak, jelikož může vypršet platnost session a proto se tokeny taky nemusí shodovat. V případě řešené aplikace byl zvolen plugin už dříve vytvořený a k lepšímu zabezpečení proti CSRF útoku jsou přidány bezpečností prvky jako přidání náhodného řetězce k heslu a následné použití hashovací funkce pro horší dešifrovatelnost, vyžadování starého hesla při změně na heslo nové, rozdělení přístupových práv mezi správce a uživatelé, ošetření vstupů od uživatelů apod.

5. Zhodnocení navrhovaného řešení

Majetkoprávní odbor má povinnost archivovat uzavřené spisy, které obsahují dokumenty zabývající se zřizováním věcných břemen, uzavíráním pronájmů, převodem a nabýváním majetku apod. po dobu pěti až deseti let. Z požadavků zaměstnanců majetkoprávního odboru MěÚ vznikla poptávka po aplikaci, umožňující rychlý přístup k dříve uzavřeným smlouvám, schváleným zastupitelstvem a radou města usnesením, uzavřeným majetkoprávním spisům a s nimi spojenými údaji, jako např. záznamy o nemovitostech, osobách vyskytujících se ve smlouvách apod.

Pro uskutečnění aplikace byl z mnoha programovacích jazyků vybrán jazyk PHP se Zend Frameworkem a databázový systém MySQL. Domnívám se, že toto řešení bylo zvoleno správně, jelikož práce v jazyce PHP je značně rychlejší než v porovnání s ostatními jazyky, ale na druhou stranu je třeba řešit množství vzniknutých chyb, díky slabému typování, na rozdíl od jazyků s typováním silným. Jazyk PHP s databázovým systémem MySQL splnil požadavky výběru a bylo v něm umožněno danou aplikaci vytvořit.

Výsledná aplikace splnila všechny požadavky zaměstnanců, a proto byl cíl práce naplněn. Předpokládám, že aplikace by mohla mít v budoucnu značný význam pro majetkoprávní odbor. Procesy a funkce v aplikaci by měly zajišťovat konzistentní a bezpečná data. Zabezpečení s největší pravděpodobností ještě bude testováno odborníky z oddělení informatiky, ale domnívám se, že základní úroveň zabezpečení v rámci aplikace je zvládnuta. Na druhou stranu aplikaci nebylo nutno zabezpečovat aplikaci protokolem HTTPS se SSL šifrováním, jelikož výsledná aplikace by měla být nasazená v rámci intranetu na Městském úřadu v Českém Těšíně.

V současné době se rozhoduje, zda a kdy bude řešená aplikace nasazena do ostrého provozu. V rámci aplikace jsou pro inovace otevřené dveře a kdykoli v budoucnu je možno danou aplikaci inovovat a rozšiřovat její funkčnost. Domnívám se, že aplikace je schopná nasazení a provozu v intranetu na MěÚ.

6. Závěr

Cílem bakalářské práce bylo navrhnout a realizovat intranetovou aplikaci pro správu majetku majetkoprávního odboru na Městském úřadě v Českém Těšíně. Aplikace měla pomáhat zaměstnancům v evidenci všech typů smluv a usnesení a s tím spojených údajů, např. nemovitostí, osob apod. Systém měl poskytovat správu úkolů s jejich termíny a možnost jak psaní svých osobních úkolů, tak psaní úkolu od správce pro zaměstnance. Jako vstup pro smlouvy a usnesení měl být prostý text a soubor přílohy. Výstupy naopak ve formě dokumentů PDF nebo uživatelem nahraných souborů. Dílčím cílem bylo umožnění běhu aplikace pouze lokálně, konkrétně na intranetu. Dalším parciálním cílem bylo zajištění vhodné úrovně bezpečnosti v aplikaci. Celkovým obecným cílem práce bylo tedy umožnění inovace ze starého typu udržování informací k novému, modernímu typu správy informací o městském majetku.

V rámci bakalářské práce byla probrána všechna teoretická východiska použitých technologií v aplikaci, včetně historických vývojů a aktualit. V teoretické části je blíže přiblížený programovací jazyk PHP se Zend Frameworkem a databázový systém MySQL. V další části se nachází analýza činností zaměstnanců majetkoprávního oboru, požadavky na výslednou aplikaci a jednotlivé možnosti oddělení informatiky. Další částí je návrh a realizace intranetové aplikace. Část návrhu je zaměřená na konzistenci aplikace a popis vývoje vnitřních procesů. Realizace je rozdělená na části podle MVC architektury, ale obsahuje i optimalizaci a zabezpečení. V poslední části této práce jsou zmíněny přínosy a nedostatky navrhovaného řešení v rámci zvolených technologií a naplnění dílčích cílů práce.

Cíl práce byl naplněn realizací intranetové aplikace, která dokáže umožnit přechod ze starých postupů správy informací k modernizované správě informací o majetku města. Výsledná aplikace dokáže evidovat smlouvy i usnesení a s nimi spojené další údaje, umožňuje spravovat úkoly správce i zaměstnanců, je zabezpečená a spolupracuje s formáty PDF a souborovými přílohami.

Seznam použité literatury

[1] ECCHER, Clint. *Profesionální webdesign: techniky a vzorová řešení*. Brno: CP Books, 2005. ISBN 80-251-0547-4.

[2] HAUSER, Marianne. *HTML a CSS: velká kniha řešení*. Brno: Computer Press, 2006. ISBN 80-251-1117-2.

[3] GREER, Tyson. *Intranety: principy a praxe.* Brno: Computer Press, 1999. ISBN 80-7226-135-5

[4] DYSON, Peter, Pat COLEMAN a Len GILBERT. *Intranet: plánování, výstavba, provoz*. Praha: Grada, 1998. ISBN 80-7169-670-6.

[5] TVRDÍKOVÁ, Milena. *Ekonomická revue: Řízená práce s digitálními dokumenty a jejich archivace*. Ostrava: Vysoká škola báňská - Technická univerzita, 2011, roč. 14, č. 2. ISSN 1212-3951.

[6] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[7] ŠKULTÉTY, Rastislav. *JavaScript: programujeme internetové aplikace*. 2. aktualiz. vyd. Brno: Computer Press, 2004. ISBN 80-251-0144-4.

[8] LACKO, Ľuboslav. *Ajax: hotová řešení*. Překlad Michal Brůha. Brno: Computer Press, 2008. ISBN 978-80-251-2108-5.

[9] MARGORÍN, Marián. *JQuery bez předchozích znalostí*. Brno: Computer Press, 2011. ISBN 978-80-251-3379-8.

[10] SHARP, Jonathan et al. *JQuery: kuchařka programátora*. Brno: Computer Press, 2010. ISBN 978-80-251-3152-7.

[11] PONKRÁC, Miloslav. *PHP a MySQL bez předchozích znalostí.* Brno: Computer Press, 2007. ISBN 978-80-251-1758-3.

[12] LAVIN, Peter. *PHP - objektově orientované: koncepty, techniky a kód*. Praha: Grada, 2009. ISBN 978-80-247-2137-8.

[13] GILMORE, Jason W. *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner Press, 2007. ISBN 978-80-86815-53-4.

[14] NARAMORE, Elizabeth et al. *PHP5, MySQL, Apache: vytváříme webové aplikace*. Brno: Computer Press, 2006. ISBN 80-251-1073-7.

[15] LECKY-THOMPSON, Ed a Steven D NOWICKI. *PHP 6: programujeme profesionálně*. Překlad Ondřej Gibl. Brno: Computer Press, 2010. ISBN 978-80-251-3127-5.

[16] SCHNEIDER, Robert D. *MySQL: oficiální průvodce tvorbou, správou a laděním databází*. Praha: Grada Publishing, 2006. ISBN 80-247-1516-3.

[17] BÖHMER, Marian. *Zend Framework: Programujeme webové aplikace v PHP*. Brno: Computer Press, 2010. ISBN 978-80-251-2965-4.

[18] ZERVAAS, Quentin. *Practical Web 2.0 applications with PHP*. New York: Apress, 2008. ISBN 978-1-59059-906-8.

[19] PADILLA, Armando. *Beginning Zend Framework*. New York: Apress, 2009. ISBN 978-1-4302-1825-8.

[20] BÖCK, Heiko. *Platforma NetBeans: podrobný průvodce programátora*. Brno: Computer Press, 2010. ISBN 978-80-251-3116-9.

[21] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Překlad Bogdan Kiszka. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

Seznam zkratek

AJAX – Asynchronous JavaScript and XML

APC – Alternative PHP Cache

ASP – Active Server Pages

CSRF – Cross-site request forgery

CSS – Cascading Style Sheets

DOM – Document Object Model

DTD – Document Type Definition

GPL – General Public License

HTML – Hypertext Markup Language

HTTP(s) – HyperText Transfer Protocol (Secure)

IČO – Identifikační číslo organizace

IDE – Integrated Development Environment

ISAM – Indexed Sequential Access Method

ISO – International Organization for Standardization

JS – JavaScript

JSP – JavaServer Pages

LAN – Local Area Network

MěÚ – Městský úřad

MIME – Multipurpose Internet Mail Extensions

MIT – Massachusetts Institute of Technology

MVC – Model-View-Controller

OOP – Objektově orientované programování

OSI - Open Systems Interconnection

PDF – Portable Document Format

PHP – Hypertext Preprocessor

PHTML – PHP skript s HTML výstupem

RSŘBD – Relační systém řízení báze dat

SQL – Structured Query Language

SSL – Secure Sockets Layer

TCP/IP – Transmission Control Protocol / Internet Protocol

URL – Uniform Resource Locator

WWW – World Wide Web

XHTML – Extensible Hypertext Markup Language

XML – Extensible Markup Language

XSS – Cross-site scripting

ZF – Zend Framework

Prohlášení o využití výsledků bakalářské práce

Prohlašuji, že

- jsem byl(a) seznámen(a) s tím, že na mou diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou (bakalářskou) práci užít (§ 35 odst. 3);
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že bibliografické údaje o diplomové (bakalářské) práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou (bakalářskou) práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 8. 5. 2013

.....

jméno a příjmení studenta

Seznam příloh

Příloha č. 1: E-R diagram

Příloha č. 2: Specifikace domén

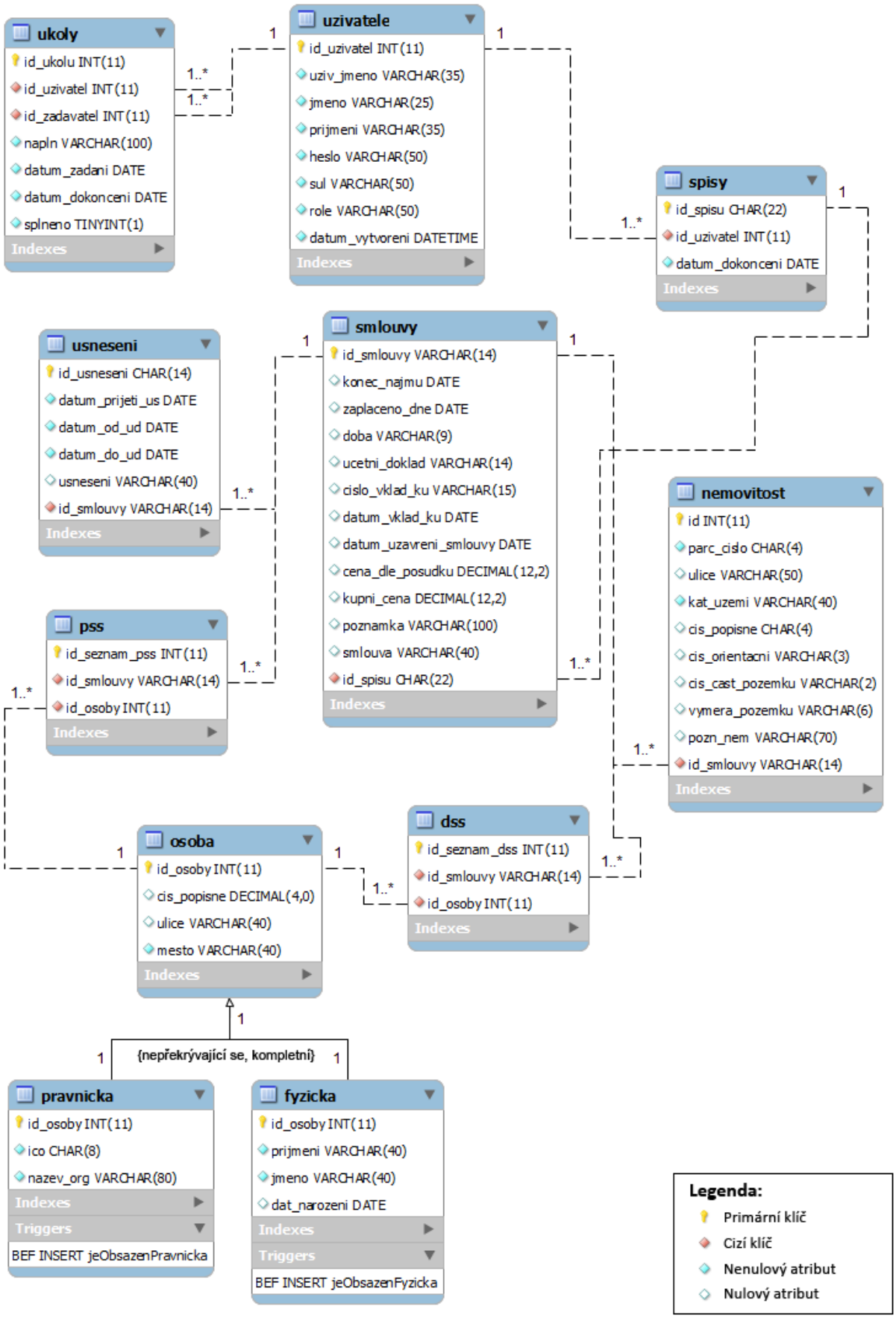
Příloha č. 3: Trigger fyzických osob a výsledek nesprávného vložení

Příloha č. 4: CLI Tool ve Windows PowerShell

Příloha č. 5: Stránka „usnesení“ a příslušné akce

Příloha č. 6 (na CD): Webová aplikace v jazyce PHP s vyexportovanou databází MySQL a přístupovými právy

Příloha č. 1: E-R diagram

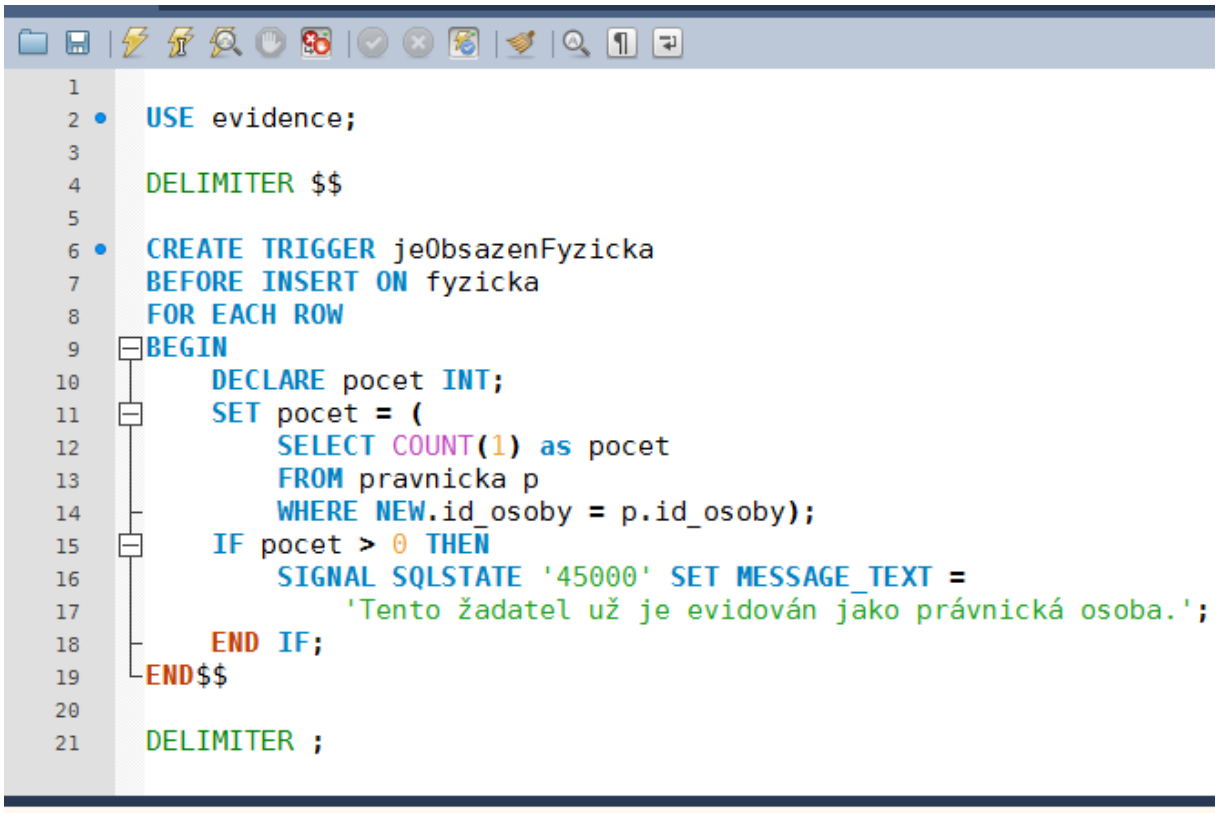


Příloha č. 2: Specifikace domén

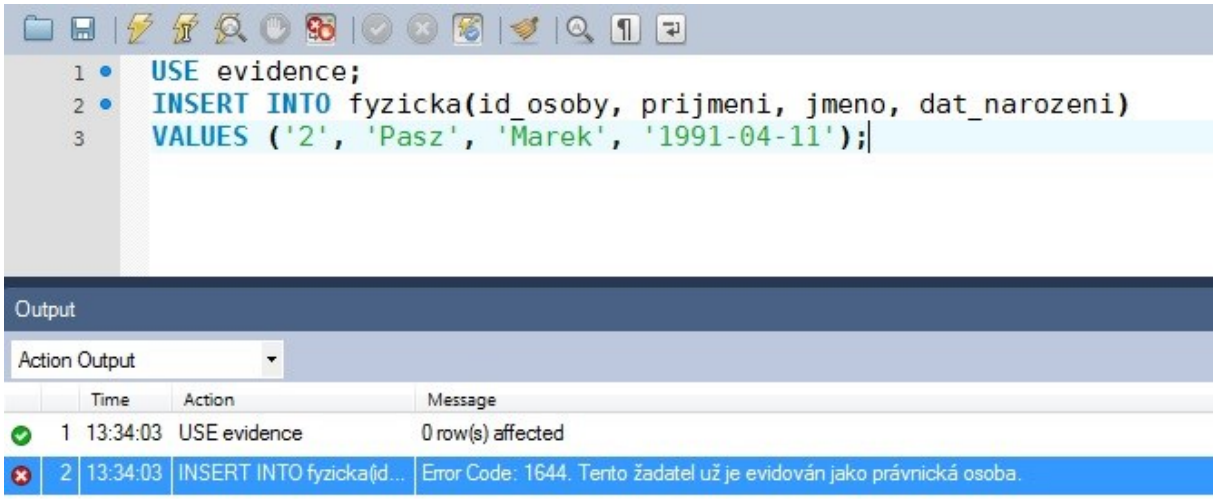
Název	Datový typ	Nulový	Popis
Ukoly			
id_ukolu #	int(11)	Ne	Identifikátor úkolu
id_uzivatel |fk|	int(11)	Ne	Identifikátor uživatele
id_zadavatel |fk|	int(11)	Ne	Identifikátor zavatele
napln	varchar(100)	Ne	Náplň úkolu
datum_zadani	date	Ne	Datum zadání
datum_dokonceni	date	Ne	Datum dokončení
splneno	tinyint(1)	Ne	Úkol splněn
Uzivatele			
id_uzivatel #	int(11)	Ne	Identifikátor uživatele
uziv_jmeno	varchar(35)	Ne	Uživatelské jméno
jmeno	varchar(25)	Ne	Jméno
prijmeni	varchar(35)	Ne	Příjmení
heslo	varchar(50)	Ne	Heslo
sul	varchar(50)	Ne	Kód pro silnější šifrování hesla
role	varchar(50)	Ne	Uživatelská role
datum_vytvoreni	datetime	Ne	Datum vytvoření uživatele
Spisy			
id_spisu #	char(22)	Ne	Identifikátor spisu
id_uzivatel	int(11)	Ne	Identifikátor uživatele
datum_dokonceni	date	Ne	Datum dokončení spisu
Smlouvy			
id_smlouvy #	varchar(14)	Ne	Identifikátor smlouvy
konec_najmu	date	Ano	Datum ukončení najmu
zaplaceno_dne	date	Ano	Den zaplacení nájmu
doba	varchar(9)	Ano	Doba nájmu
ucetni_doklad	varchar(14)	Ano	Účetní doklad
cislo_vklad_ku	varchar(15)	Ano	Číslo vkladu na katastrální úřad
datum_vklad_ku	date	Ano	Datum vkladu na katastrální úřad
datum_uzavreni_smlouvy	date	Ano	Datum uzavření smlouvy
cena_dle_posudku	decimal(12,2)	Ano	Cena dle znaleckého posudku
kupni_cena	decimal(12,2)	Ano	Kupní cena
poznamka	varchar(100)	Ano	Poznámka
smlouva	varchar(40)	Ano	Cesta k souboru smlouvy
id_spisu |fk|	char(22)	Ne	Identifikátor spisu
Usneseni			
id_usneseni #	char(14)	Ne	Identifikátor usnesení
datum_prijeti_us	date	Ne	Datum schválení
datum_od_ud	date	Ne	Zvěřejněno na úřední desce
datum_do_ud	date	Ne	Sňato z úřední desky

Název	Datový typ	Nulový	Popis
usneseni	varchar(40)	Ano	Cesta k souboru usesení
id_smlouvy |fk|	varchar(14)	Ne	Identifikátor smlouvy
Nemovitost			
id #	int(11)	Ne	Identifikátor nemovitosti
parc_cislo	char(4)	Ne	Číslo parcely
ulice	varchar(50)	Ano	Ulice
kat_uzemi	varchar(40)	Ne	Katastrální území
cis_popisne	char(4)	Ano	Číslo popisné
cis_orientacni	varchar(3)	Ano	Číslo orientační
cis_cast_pozemku	varchar(2)	Ano	Číslo části pozemku
vymera_pozemku	varchar(6)	Ano	Výměra pozemku
pozn_nem	varchar(70)	Ano	Poznámka k nemovitosti
id_smlouvy |fk|	varchar(14)	Ne	Identifikátor smlouvy
Pss (První smluvní strany)			
id_seznam_pss #	int(11)	Ne	Identifikátor první smluvní strany
id_smlouvy |fk|	varchar(14)	Ne	Identifikátor smlouvy
id_osoby |fk|	int(11)	Ne	Identifikátor osoby
Dss (Druhé smluvní strany)			
id_seznam_dss #	int(11)	Ne	Identifikátor druhé smluvní strany
id_smlouvy |fk|	varchar(14)	Ne	Identifikátor smlouvy
id_osoby |fk|	int(11)	Ne	Identifikátor osoby
Osoba			
id_osoby #	int(11)	Ne	Identifikátor osoby
cis_popisne	decimal(4,0)	Ano	Číslo popisné
ulice	varchar(40)	Ano	Ulice
mesto	varchar(40)	Ne	Město
Fyzicka			
id_osoby # |fk|	int(11)	Ne	Identifikátor osoby
prijmeni	varchar(40)	Ne	Příjmení
jmeno	varchar(40)	Ne	Jméno
dat_narozeni	date	Ano	Datum narození
Pravnicka			
id_osoby # |fk|	int(11)	Ne	Identifikátor osoby
ico	char(8)	Ne	Identifikační číslo organizace
nazev_org	varchar(80)	Ne	Název organizace

Příloha č. 3: Trigger fyzických osob a výsledek nesprávného vložení



```
1
2  USE evidence;
3
4  DELIMITER $$
5
6  CREATE TRIGGER jeObsazenFyzicka
7  BEFORE INSERT ON fyzicka
8  FOR EACH ROW
9  BEGIN
10     DECLARE pocet INT;
11     SET pocet = (
12         SELECT COUNT(1) as pocet
13         FROM pravnicka p
14         WHERE NEW.id_osoby = p.id_osoby);
15     IF pocet > 0 THEN
16         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
17             'Tento žadatel už je evidován jako právnická osoba.';
18     END IF;
19 END$$
20
21 DELIMITER ;
```



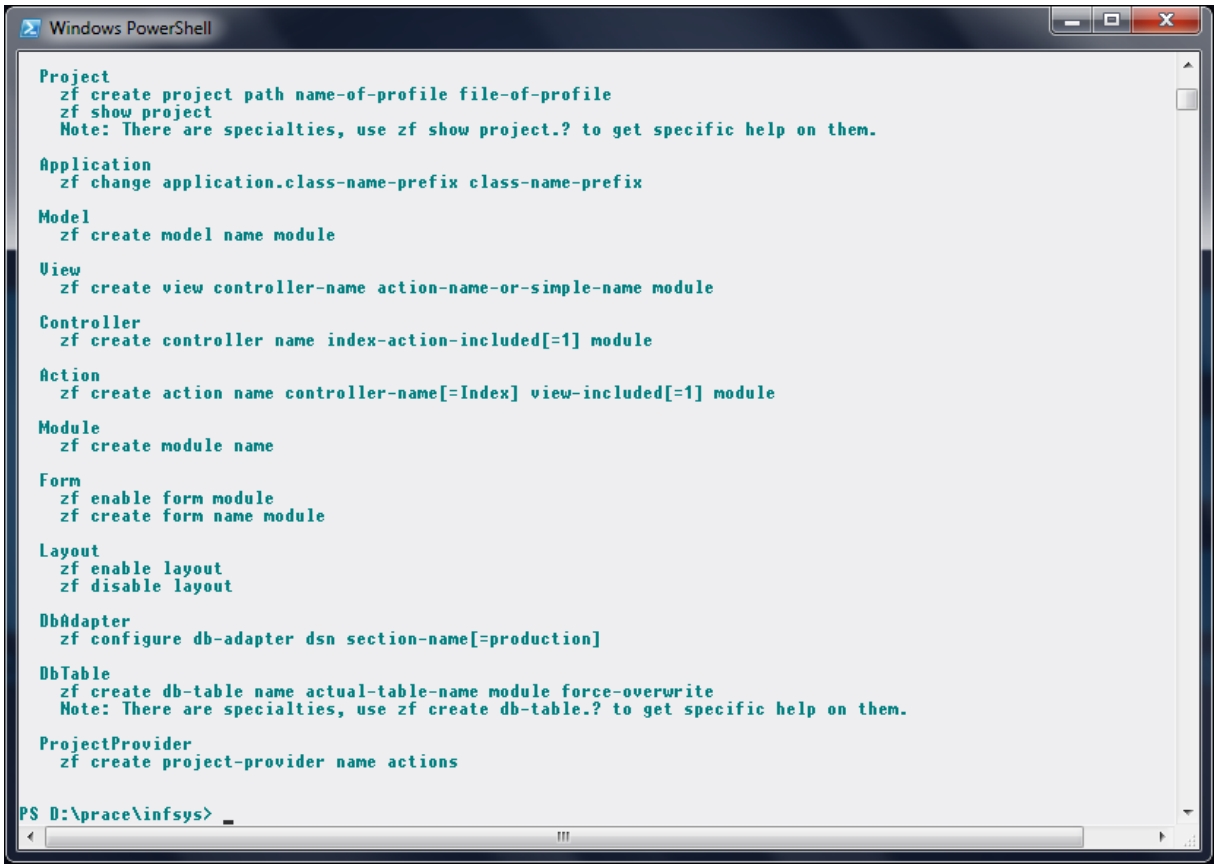
```
1  USE evidence;
2  INSERT INTO fyzicka(id_osoby, prijmeni, jmeno, dat_narozeni)
3  VALUES ('2', 'Pasz', 'Marek', '1991-04-11');
```

Output

Action Output

	Time	Action	Message
1	13:34:03	USE evidence	0 row(s) affected
2	13:34:03	INSERT INTO fyzicka(id...	Error Code: 1644. Tento žadatel už je evidován jako právnická osoba.

Příloha č. 4: CLI Tool ve Windows PowerShell



Příloha č. 5: Stránka „usnesení“ a příslušné akce